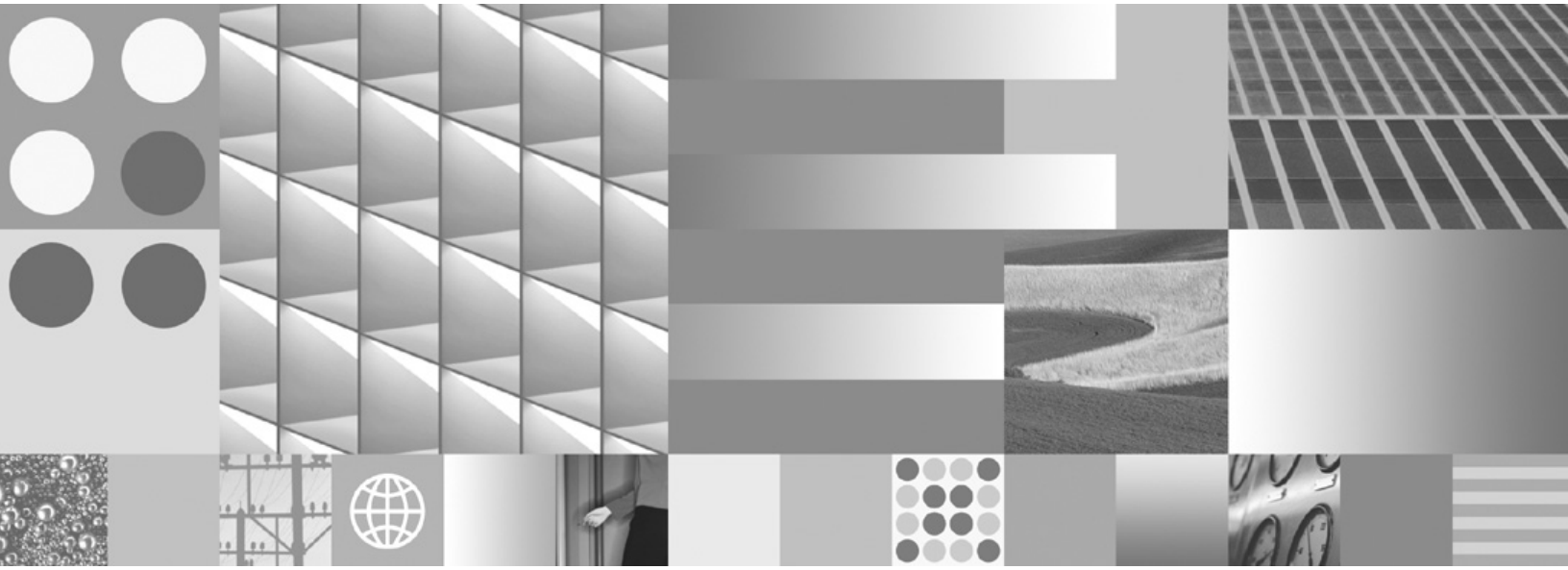
DB2 版本 9.5
Linux 版、UNIX 版和 Windows 版

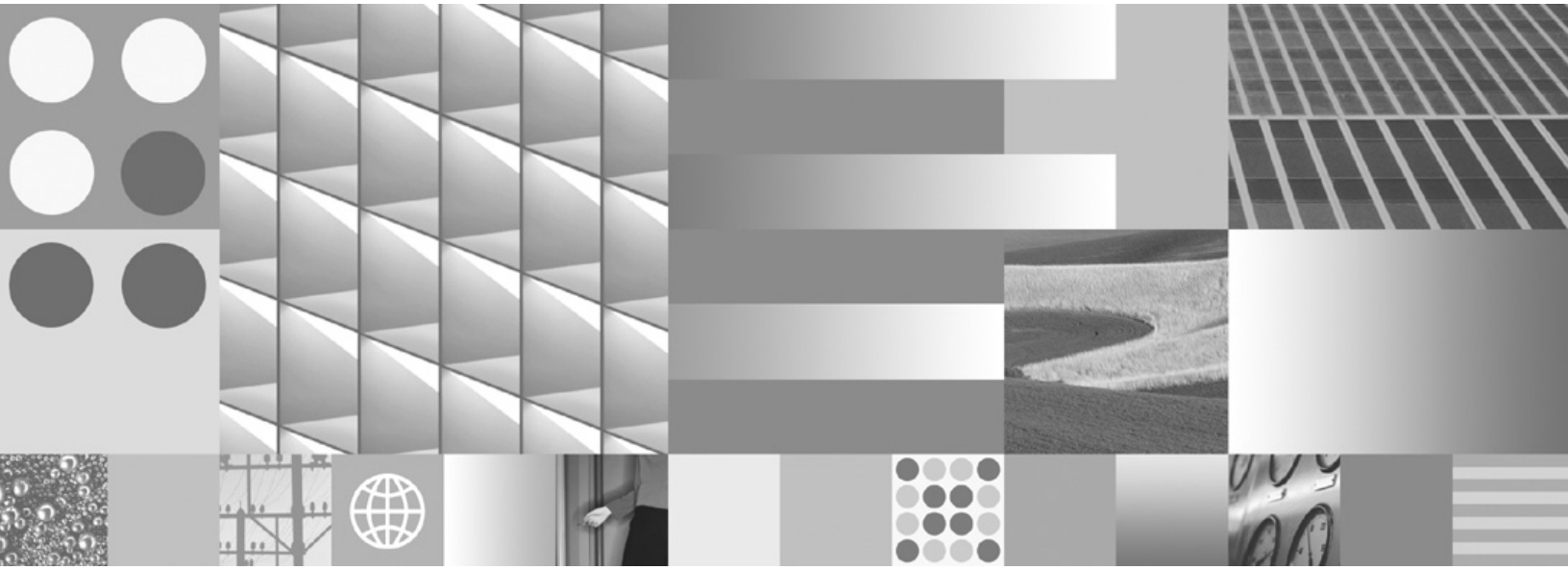


数据移动指南和参考

2008 年 3 月更新

S151-0617-01

DB2 版本 9.5
Linux 版、UNIX 版和 Windows 版



数据移动指南和参考

2008 年 3 月更新

S151-0617-01

注意

使用此信息及其支持的产品前，请先阅读第 413 页的附录 F, 『声明』下的常规信息。

修订版声明

此文档包含 IBM 的所有权信息。它在许可协议中提供，且受版权法的保护。本出版物中包含的信息不包括对任何产品的保证，且提供的任何语句都不需要如此解释。

您可在线或通过当地的 IBM 代表处订购 IBM 出版物。

- 要在线订购出版物，请转至 IBM 出版物中心，网址为: www.ibm.com/shop/publications/order
- 要查找当地的 IBM 代表处，请转至 IBM 全球联系人目录，网址为: www.ibm.com/planetwide

要从美国或加拿大的 DB2 市场和销售部订购 DB2 出版物，请致电 1-800-IBM-4YOU（426-4968）。

当您向 IBM 发送信息时，即同意授予 IBM 独一无二的权力以它认为适当且不会对您造成任何影响的方式使用或分发该信息。

目录

关于本书

本书提供了有关以下DB2® 数据库 Linux® 版、UNIX® 版和 Windows® 版数据移动实用程序的信息，并演示了如何使用这些实用程序:

- Export 实用程序和 Import 实用程序

  export 实用程序和 import 实用程序使用 DB2 Connect™ 在表或视图以及其他数据库或电子表格程序之间、在 DB2 数据库之间，以及在DB2 数据库和主机数据库之间移动数据。 export 实用程序将数据库中的数据移到操作系统文件中，您可以使用这些文件将那些数据导入或装入到另一个数据库中。

- Load 实用程序

  load 实用程序将数据移到表中，扩展现有的索引，并生成统计信息。移动大量数据时，load 实用程序的速度比 import 实用程序快得多。使用 export 实用程序导出的数据可使用 load 使用程序装入。

  在分区数据库环境中使用 load 实用程序时，可分布大量数据并将它们装人到另一个数据库分区中。

有关数据移动选项的完整列表的信息，请参阅 DB2 V9.5 中可用的的数据移动选项。

第 1 部分 数据移动实用程序和参考

第 1 章 DB2 V9.5 中提供的数据移动选项

DB2 V9.5 中提供了各种数据移动选项。下表概述了您可以使用的数据移动工具和实用程序。此表可帮助您确定哪些数据移动选项能最好地满足您的需求。

表 1. DB2 V9.5 中提供的数据移动选项

实用程序名称	**load 实用程序**
目的	高效地将大量数据移到新创建的表或者已包含数据的表中。
是否跨平台兼容	是
最佳实践用法	当性能是您关注的主要方面时，使用此实用程序最合适。此实用程序可以用作 import 实用程序的备用实用程序。由于它直接将格式化的页写入数据库而不必执行 SQL INSERTS，所以它的速度比 import 实用程序快。此外，load 实用程序还允许您选择不记录数据或使用 COPY 选项保存已装入数据的副本。装入操作可以充分利用 SMP 和 MPP 环境中的资源，如 CPU 和内存。
参考资料	装入数据

实用程序名称	**db2move**
目的	通过将 db2move 实用程序与 COPY 选项配合使用，您可以将模式模板（包含数据或不包含数据）从源数据库复制到目标数据库，或者将整个模式从源数据库移动到目标数据库。将 db2move 实用程序与 IMPORT 或 EXPORT 选项配合使用有利于在 DB2 数据库之间移动大量表。
是否跨平台兼容	是
最佳实践用法	与 COPY 选项配合使用时，源数据库和目标数据库必须不同。COPY 选项对于生成模式模板很有用。当跨平台备份和复原操作不受支持时，可使用 IMPORT 或 EXPORT 选项来克隆数据库。IMPORT 和 EXPORT 选项将与 db2look 命令配合使用。
参考资料	• *《数据服务器、数据库和数据库对象指南》*中的『复制模式』 • 已导入表重新创建

实用程序名称	**import 实用程序**
目的	将外部文件中的数据插入到表、层次结构、视图或昵称中
是否跨平台兼容	是

实用程序名称	**import 实用程序**
最佳实践用法	在下列情况下，import 实用程序可作为 load 实用程序的较好备用实用程序: • 目标表是视图 • 目标表具有约束，并且您不想让目标表处于“设置完整性暂挂”状态 • 目标表具有触发器，并且您希望将其触发
参考资料	导入数据

实用程序名称	**export 实用程序**
目的	以其中一种外部文件格式导出数据库中的数据。稍后可以导入或装入这些数据。
是否跨平台兼容	是
最佳实践用法	当您希望将数据存储在外部文件中以进行进一步处理或以便将数据移到另一个表时，使用此实用程序最合适。High Performance Unload（HPU）是此实用程序的备用方法，但必须单独购买此产品。export 实用程序支持 XML 列。
参考资料	导出数据

实用程序名称	**ADMIN_COPY_SCHEMA 过程**
目的	允许您生成单个模式中所有对象的副本，然后在新模式中重新创建这些对象。可以在数据库中使用数据或不使用数据执行此复制操作。
是否跨平台兼容	是
最佳实践用法	此实用程序对于生成模式模板很有用。如果您想要试验模式（例如，尝试使用新索引），但不想影响源模式的行为，那么此实用程序也很有用。ADMIN_COPY_SCHEMA 过程与 db2move 实用程序的主要差别在于: • ADMIN_COPY_SCHEMA 过程用于单个数据库，而 db2move 实用程序在各个数据库之间使用 • 如果在调用 db2move 实用程序时它不能创建物理对象（如表或索引），那么此实用程序将失败。ADMIN_COPY_SCHEMA 过程将记录错误并继续。 • ADMIN_COPY_SCHEMA 过程使用“从游标装入”将数据从一个模式移到另一个模式。db2move 实用程序使用远程装入（类似于“从游标装入”），此操作从源数据库中拉出数据。
参考资料	*《数据服务器、数据库和数据库对象指南》*中的『复制模式』

实用程序名称	**带有 REDIRECT 选项和 GENERATE SCRIPT 选项的 Restore 实用程序**
目的	使用现有备份映像中的脚本将整个数据库从一个系统复制到另一个系统。
是否跨平台兼容	受限制。请参阅“参考资料”。
最佳实践用法	当存在备份映像时，使用此实用程序最合适。
参考资料	• *《数据恢复及高可用性指南与参考》*中的『使用自动生成的脚本执行重定向复原』 • *《数据恢复及高可用性指南与参考》*中的『不同操作系统和硬件平台之间的备份与复原操作』

实用程序名称	**db2relocatedb - 重新分配数据库命令**
目的	重命名数据库，或将整个或部分数据库重新分配给同一系统或另一系统。
是否跨平台兼容	否
最佳实践用法	• 当 Backup 和 Restore 实用程序要耗用大量时间时，可以使用此实用程序。 • 在移动或创建数据库副本时，可将此实用程序用作 Backup 和 Restore 实用程序的备用实用程序。 • 此实用程序还提供了一种快速方法来为其他环境（如测试）克隆数据库。 • 可以使用此实用程序将表空间容器移到一组新的存储设备。
参考资料	*Command Reference*中的『db2relocatedb - 重定位数据库命令』

实用程序名称	**分割镜像**
目的	创建克隆、备用或备份数据库。
是否跨平台兼容	否
最佳实践用法	• 创建备用系统，以便在主系统发生故障时减少停机时间 • 在分割数据库上执行备份操作，而不用在实时生产机器上执行此操作 • 提供一种快速方法来为其他环境（如测试）克隆数据库
注意事项	• 只能在数据库的分割版本上备份 DMS 表空间 • 通常与存储系统提供的瞬间复制技术一起使用 • 备用方法是在数据库暂挂后立即发出文件副本，但这样会使用于数据库的存储器空间加倍

实用程序名称	分割镜像
参考资料	*《数据恢复及高可用性指南与参考》*中的『通过联机分割镜像和暂挂 I/O 支持实现高可用性』

第 2 章 export 实用程序

export 实用程序概述

export 实用程序会使用 SQL select 语句或 XQuery 语句抽取数据，并将该信息放到文件中。您可使用输出文件移动数据以便将来执行导入或装入操作，或者将数据用于分析。

export 实用程序是一个相对简单而且具有灵活性操作的数据移动实用程序。 可通过下列方法激活它：通过控制中心、在 CLP 中发出 EXPORT 命令、调用 ADMIN_CMD 存储过程或通过应用程序调用 db2Export API。

下列各项是基本导出操作所必需的:

- 要用于存储已导出数据的操作系统文件的路径和名称
- 输入文件中的数据格式
  Export 支持对输出文件使用 IXF、WSF 和 DEL 数据格式。
- 指定要导出的数据
  对于大部分导出操作，您需要提供 SELECT 语句指定需要进行检索以便导出的数据。导出类型表时，不必显式发出 SELECT 语句；您只需要指定层次结构中的子表遍历顺序

如果需要以 IXF 格式移动数据，可将 export 实用程序与 DB2 Connect 配合使用。

其他选项

许多参数允许您定制导出操作。文件类型修饰符提供了允许您更改数据、日期和时间戳记或代码页格式之类的许多选项，或者已编写特定数据类型以分隔文件。通过使用 **METHOD** 参数，可指定要用于已导出数据的不同列名。

可从包括一个或多个 XML 数据类型列的表中导出数据。使用 **XMLFILE**、**XML TO** 和 **XMLSAVESCHEMA** 参数指定有关如何存储已导出文档的详细信息。

有几种方法可用来改进 export 实用程序的性能。因为 export 实用程序是嵌入式 SQL 应用程序并且以内部方式执行 SQL 访存，所以应用于 SQL 操作的优化会同时应用于 export 实用程序。请考虑采用大型缓冲池、建立索引和排序堆的好处。此外，请尝试通过将输出文件放在容器和日志设备外部来尽量降低输出文件争用问题。

消息文件

export 实用程序会将错误消息、警告消息和参考消息写至标准 ASCII 文本消息文件。对于 CLP 以外的所有接口，必须预先使用 **MESSAGES** 参数指定这些文件的名称。如果要使用 CLP 并且不指定消息文件，那么 export 实用程序会将消息写至标准输出。

使用 export 实用程序所需的特权和权限

特权允许您创建、更新、删除或访问数据库资源。权限级别提供了一种方法，以允许您将特权映射至高级数据库管理器维护和实用程序操作。

特权和权限共同控制数据库管理器及其数据库对象的访问。用户只能访问他们对其具有相应权限（即必需的特权或权限）的对象。

对于参与导出操作的每个表或视图，您必须具有 SYSADM 或 DBADM 权限或 CONTROL 或 SELECT 特权。

导出 LBAC 保护的数据时，必须允许会话授权标识读取您尝试导出的行或列。会话授权标识无权读取的受保护行不会导出。如果 SELECT 语句包括不允许会话授权标识读取的任何受保护列，那么 export 实用程序会失败，并且返回错误（SQLSTATE 42512）。

导出数据

使用 export 实用程序将数据从数据库导出至文件。该文件可使用若干外部文件格式中的一种。可以通过提供 SQL SELECT 语句或类型表分层信息来指定要导出的数据。

您需要对应每个参与表或视图的 SYSADM 权限、DBADM 权限、CONTROL 特权或 SELECT 特权才能从数据库导出数据

在运行 export 实用程序之前，必须连接或能够隐式连接至要从中导出数据的数据库。如果启用了隐式连接，那么将建立与缺省数据库的连接。实用程序必须借助通过引擎（而不是通过 DB2 Connect 网关或回送环境）的直接连接从 Linux、UNIX 或 Windows 客户机访问 Linux、UNIX 或 Windows 数据库服务器。

因为实用程序会发出 COMMIT 语句，所以应在运行 export 实用程序之前发出 COMMIT 或 ROLLBACK 语句来完成所有事务并释放所有锁定。访问表并使用独立连接的应用程序不必断开连接。

export 实用程序有下列限制:

- 此实用程序不支持带有结构化类型列的表。

可使用下列方式运行 export 实用程序: 使用控制中心中的“导出表”笔记本、使用应用程序编程接口（API）db2Export 或在命令行处理器（CLP）中指定 EXPORT 命令。

使用“导出表”笔记本

要使用“导出表”笔记本导出数据:

1. 在“控制中心”中，展开对象树，直到找到“表”或“视图”文件夹为止。
2. 单击要使用的文件夹。任何现有的表或视图都将显示在窗口右边的窗格（内容窗格）中。
3. 在内容窗格中，右键单击想要的表或视图，然后从弹出菜单中选择“导出”。“导出表”笔记本将打开。

“控制中心”联机帮助工具中提供了有关“导出表”笔记本的详细信息。

通过使用 CLP 发出 EXPORT 命令

这是一个非常简单的导出操作，只需要您对 SELECT 语句指定目标文件、文件格式和源文件。

要从 CLP 中导出数据，请输入 EXPORT 命令:

```
db2 export to filename of ixf select * from table
```

其中 filename 是要创建并导出的输出文件的名称，ixf 是文件格式，而 table 是包含要复制的数据的表名。

但是，您可能还想指定用于写入警告消息和错误消息的消息文件。为此，添加 **MESSAGES** 参数和消息文件名称（在此情况下为 msg.txt），所以命令为如下所示:

```
db2 export to filename of ixf messages msgs.txt select * from table
```

有关完整语法和用法信息，请参阅“EXPORT 命令”。

导出 XML 数据

导出 XML 数据时，生成的 QDM（XQuery 数据模型）实例将写入与包含导出的关系数据的主数据文件不同的文件。即使未指定 XMLFILE 和 XML TO 选项亦如此。缺省情况下，导出的 QDM 实例将全部放入同一个 XML 文件中。可以使用 XMLINSEPFILES 文件类型修饰符来指定将每个 QDM 实例写入不同文件。

然而，主数据文件中用 XML 数据说明符（XDS）表示 XML 数据。XDS 是表示为 XML 标记（其名称是“XDS”）的字符串，它具有用于描述关于列中实际 XML 数据的信息的属性；这种信息涉及包含实际 XML 数据的文件名，以及该文件内 XML 数据的偏移量和长度。

可以使用 XML TO 和 XMLFILE 选项指定导出的 XML 文件的目标路径和基本名称。如果指定了 XML TO 或 XMLFILE 选项，那么已导出 XML 文件名的格式（存储在 XDS 的 FIL 属性中）为 xmlfilespec.xxx.xml，其中 xmlfilespec 是对 XMLFILE 选项指定的值，而 xxx 是 EXPORT 实用程序生成的 XML 文件的序号。否则，已导出 XML 文件名的格式为 exportfilename.xxx.xml，其中 exportfilename 是对 EXPORT 命令指定的已导出输出文件的名称，而 xxx 是 EXPORT 实用程序生成的 XML 文件的序号。

缺省情况下，导出的 XML 文件将写入已导出数据文件的路径中。导出的 XML 文件的缺省基本名称包括已导出数据文件的名称、追加的 3 位序号和 .xml 扩展名。

示例

在下列示例中，假定 USER.T1 表包含四列两行:

```
C1 INTEGER
C2 XML
C3 VARCHAR(10)
C4 XML
```

表 2. USER.T1

C1	C2	C3	C4
2	<?xml version=″1.0″ encoding=″UTF-8″ ?><note time=″12:00:00″><to>You</to><from> Me</from><heading>note1</heading> <body>Hello World!</body></note>	’char1’	<?xml version=″1.0″ encoding=″UTF-8″ ?><note time=″13:00:00″><to>Him</to><from> Her</from><heading>note2</heading>< body>Hello World!</body></note>
4	NULL	’char2’	?xml version=″1.0″ encoding=″UTF-8″ ?><note time=″14:00:00″>to>Us</to><from> Them</from><heading>note3</heading> <body>Hello World!</body></note>

示例 1

以下命令将定界 ASCII（DEL）格式的 USER.T1 的内容导出到“/mypath/t1export.del”文件中。因为没有指定 XML TO 和 XMLFILE 选项，所以将 C2 和 C4 列中包含的 XML 文档与导出的主文件“/mypath”写入同一路径中。这些文件的基本名称为“t1export.del.xml”。XMLSAVESCHEMA 选项指示将在导出过程中保存 XML 模式信息。

```
EXPORT TO /mypath/t1export.del OF DEL XMLSAVESCHEMA SELECT * FROM USER.T1
```

导出的文件“/mypath/t1export.del”包含:

```
2,"<XDS FIL='t1export.del.001.xml' OFF='0' LEN='144' />","char1",
"<XDS FIL='t1export.del.001.xml' OFF='144' LEN='145' />"
4,,"char2","<XDS FIL='t1export.del.001.xml' OFF='289'
LEN='145' SCH='S1.SCHEMA_A' />"
```

导出的 XML 文件“/mypath/t1export.del.001.xml”包含:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
   <from>Me</from><heading>note1</heading><body>Hello World!</body>
   </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00"><to>Him
   </to><from>Her</from><heading>note2</heading><body>Hello World!
   </body></note><?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00">
   <to>Us</to><from>Them</from>heading>note3</heading><body>
   Hello World!</body></note>
```

示例 2

以下命令将 DEL 格式的 USER.T1 的内容导出到“t1export.del”文件中。将 C2 和 C4 列中包含的 XML 文档写入“/home/user/xmlpath”路径中。使用基本名称“xmldocs”命名 XML 文件，并将导出的多个 XML 文档写入同一个 XML 文件。XMLSAVESCHEMA 选项指示将在导出过程中保存 XML 模式信息。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs XMLSAVESCHEMA SELECT * FROM USER.T1
```

导出的 DEL 文件“/home/user/t1export.del”包含:

```
2,"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='144' />","char1",
"<XDS FIL='xmldocs.001.xml' OFF='144' LEN='145' />"
4,,"char2","<XDS FIL='xmldocs.001.xml' OFF='289'
LEN='145' SCH='S1.SCHEMA_A' />"
```

导出的 XML 文件“/home/user/xmlpath/xmldocs.001.xml”包含:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
   <from>Me</from><heading>note1</heading><body>Hello World!</body>
   </note><?xml version="1.0" encoding="UTF-8" ?><note time="13:00:00">
   <to>Him</to><from>Her</from><heading>note2</heading><body>
   Hello World!</body></note><?xml version="1.0" encoding="UTF-8" ?>
   <note time="14:00:00"><to>Us</to><from>Them</from><heading>
   note3</heading><body>Hello World!</body></note>
```

示例 3

除了将导出的每个 XML 文档写入不同 XML 文件外，以下命令与示例 2 类似。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs MODIFIED BY XMLINSEPFILES XMLSAVESCHEMA
SELECT * FROM USER.T1
```

导出的文件“/mypath/t1export.del”包含:

```
2,"<XDS FIL='xmldocs.001.xml' />","char1","XDS FIL='xmldocs.002.xml' />"
4,,"char2","<XDS FIL='xmldocs.004.xml' SCH='S1.SCHEMA_A' />"
```

导出的 XML 文件“/home/user/xmlpath/xmldocs.001.xml”包含:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="12:00:00"><to>You</to>
   <from>Me</from><heading>note1</heading><body>Hello World!</body>
   </note>
```

导出的 XML 文件“/home/user/xmlpath/xmldocs.002.xml”包含:

```
?xml version="1.0" encoding="UTF-8" ?>note time="13:00:00">to>Him/to>
   from>Her/from>heading>note2/heading>body>Hello World!/body>
   /note>
```

导出的 XML 文件“/home/user/xmlpath/xmldocs.004.xml”包含:

```
<?xml version="1.0" encoding="UTF-8" ?><note time="14:00:00"><to>Us</to>
   <from>Them</from><heading>note3</heading><body>Hello World!</body>
   </note>
```

示例 4

以下命令将 XQuery 的结果写入 XML 文件。

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /home/user/xmlpath
XMLFILE xmldocs MODIFIED BY XMLNODECLARATION select
xmlquery( '$m/note/from/text()' passing by ref c4 as "m" returning sequence)
   from USER.T1
```

导出的 DEL 文件“/mypath/t1export.del”包含:

```
"<XDS FIL='xmldocs.001.xml' OFF='0' LEN='3' />"
"<XDS FIL='xmldocs.001.xml' OFF='3' LEN='4' />"
```

导出的 XML 文件“/home/user/xmlpath/xmldocs.001.xml”包含:

```
HerThem
```

注: 此特定 XQuery 的结果不生成结构良好的 XML 文档。因此，不能将上面导出的文件直接导入到 XML 列中。

LBAC 保护的数据导出注意事项

导出受基于标号的访问控制（LBAC）保护的数据时，导出的数据仅限于 LBAC 凭证允许您读取的数据。

如果 LBAC 凭证不允许您读取某行，那么该行不会导出，但不会返回任何错误。如果 LBAC 凭证不允许您读取某列，那么 export 实用程序失败，并且返回错误（SQLSTATE 42512）。

数据类型为 DB2SECURITYLABEL 列中的值将作为括在字符定界符中的原始数据导出。如果原始数据中包含字符定界符，那么会使用双字符定界符。不会对构成导出值的字节进行任何其他更改。这意味着包含 DB2SECURITYLABEL 数据的数据文件可包含换行符、换页符或其他非可打印 ASCII 字符。

如果希望以便于人们阅读的格式导出数据类型为 DB2SECURITYLABEL 的列值，可在 SELECT 语句中使用 SECLABEL_TO_CHAR 标量函数以将这些值转换为安全标号字符串格式。

示例

在下列示例中，输出使用 DEL 格式，并且会写至文件 myfile.del。数据将从表 REPS 中导出，该表是使用以下语句创建的:

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

此示例会以缺省格式导出 row_label 列的值:

```
db2 export to myfile.del of del select * from reps
```

因为 row_label 列的值可能包含若干 ASCII 控制字符，所以该数据文件在大多数文本编辑器中的可读性不太好:

以下示例会以安全标号字符串格式导出 row_label 列的值:

```
db2 export to myfile.del of del select SECLABEL_TO_CHAR(row_label,
'DATA_ACCESS_POLICY'), id, name from reps
```

以下是上一示例创建的数据文件的摘录。注意，安全标号的格式是可阅读的:

```
...
"Secret:():Epsilon 37", 2005, "Susan Liu"
"Secret:():(Epsilon 37,Megaphone,Cloverleaf)", 2006, "Johnny Cogent"
"Secret:():(Megaphone,Cloverleaf)", 2007, "Ron Imron"
...
```

表导出注意事项

典型导出操作包括插入或装入到现有表中的所选数据的输出。但是，也可导出整个表，以便以后使用 import 实用程序重新创建。

要导出表，必须指定 PC/IXF 文件格式。然后可通过 CREATE 方式使用 import 实用程序以重新创建已保存表（包括其索引）。但是，如果出现下列任一情况，那么某些信息不会保存至已导出 IXF 文件:

- 索引列名包含十六进制值 0x2B 或 0x2D。
- 该表包含 XML 列。
- 该表是多维集群表（MDC）。
- 该表包含表分区键。
- 由于代码页转换，索引名长度超过 128 个字节。
- 该表是受保护的。
- EXPORT 命令包含 SELECT * FROM *tablename* 以外的操作字符串。
- 对 export 实用程序指定了 **METHOD N** 参数。

有关丢失的表属性列表，请参阅“表导入注意事项”。如果任何信息未保存，那么重新创建表时会返回警告 SQL27984W。

注: 不建议使用导入的 CREATE 方式。请使用 db2look 实用程序来捕获并重新创建表。

索引信息

如果索引中指定的列名包含 - 或 + 字符，那么不会收集索引信息，并且将返回警告 SQL27984W。export 实用程序完成处理，并且不会影响已导出的数据。但是，索引信息未保存在 IXF 文件中。因此，您必须使用 db2look 实用程序来单独创建索引。

空间局限性

如果导出的数据超过创建导出文件所在文件系统的可用空间量，那么导出操作会失败。在这种情况下，应该通过在 WHERE 子句中指定条件来对选择的数据量进行限制，以使已导出文件能够存放在目标文件系统中。可以多次调用 export 实用程序以导出所有数据。

使用其他文件格式的表

如果未使用 IXF 文件格式进行导出，那么输出文件不包含目标表的描述，但它们包含记录数据。要重新创建表及其数据，请创建目标表，然后使用 load 或 import 实用程序填充该表。可使用 db2look 实用程序来捕获原始表定义，并生成相应的数据定义语言（DDL）。

类型表导出注意事项

可使用 DB2 export 实用程序将数据移出类型表以便以后导入。通过遵循特定顺序并创建中间平面文件，导出会将数据从类型表的一个分层结构移至另一个分层结构。

处理类型表时，export 实用程序会控制输出文件中所放置的内容；仅指定目标表名和（可选）WHERE 子句。只能通过指定目标表名和 WHERE 子句来表达子查询语句。在导出层次结构时，不能指定全查询或 SELECT 语句。

使用遍历顺序保留层次结构

类型表可以在层次结构中。有几种方法可用来在层次结构之间移动数据:

- 从一个层次结构移至完全相同的层次结构
- 从一个层次结构移至更大层次结构的子节。
- 从大型层次结构的子节移至单独的层次结构

层次结构中的类型标识与数据库有关，这意味着在不同数据库中，相同类型有不同的标识。因此，在这些数据库之间移动数据时，必须对相同的类型进行映射以确保正确地移动数据。

用于类型表的映射称为*遍历顺序*，此顺序表示从上到下从左到右浏览层次结构中所有超表和子表。在导出操作期间写出每一类型行时，将把标识转换为索引值。此索引值可以是 1 到层次结构中相关类型数的任何一个数字。索引值是通过按特定顺序（遍历顺序）遍历层次结构时对每种类型进行编号生成的。图 1 显示具有 4 种有效遍历顺序的层次结构:

- Person -> Employee -> Manager -> Architect -> Student
- Person -> Student -> Employee -> Manager -> Architect
- Person -> Employee -> Architect -> Manager -> Student
- Person -> Student -> Employee -> Architect -> Manager

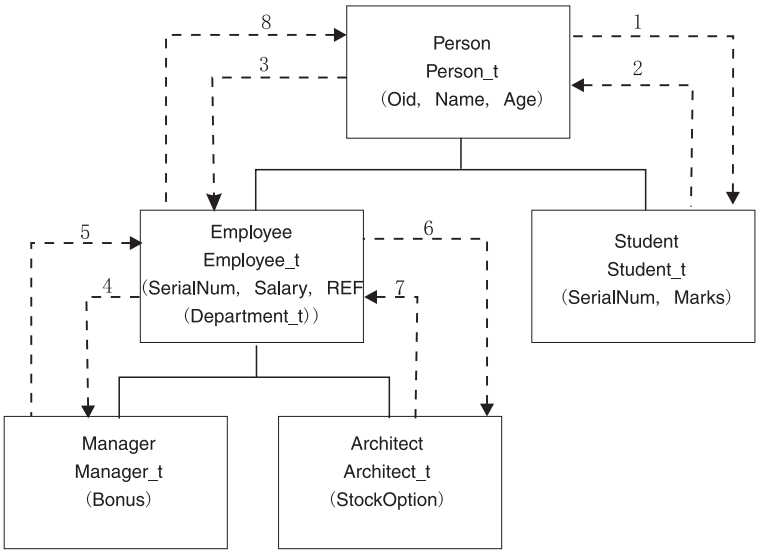


图 1. 层次结构示例

由于遍历顺序确定了一些数据相对于其他数据的移动目标位置，所以，在表层次结构之间移动数据时，此顺序十分重要。有两种类型的遍历顺序: *缺省遍历顺序*和*用户指定的遍历顺序*。

缺省遍历顺序

使用缺省遍历顺序时，所有相关类型都会引用层次结构中的所有可访问类型（从层次结构中的特定起始点开始）。缺省顺序包括层次结构中的所有表，并且每个表按 OUTER 顺序谓词中使用的方案排序。例如，图 1 中用虚线指示的缺省遍历顺序为 Person -> Student -> Employee -> Manager -> Architect。

在与不同文件格式配合使用时，缺省遍历顺序的行为有所不同。如果将数据导出为 PC/IXF 文件格式，那么将创建所有相关类型、它们的定义及相关表的记录。export 实用程序还会完成索引值对每个表进行映射的操作。当使用 PC/IXF 文件格式时，应使用缺省遍历顺序。

对于 ASC、DEL 或 WSF 文件格式，即使源层次结构和目标层次结构具有完全相同的结构，创建类型行和类型表的顺序也可能不同。这会导致缺省遍历顺序标识继续遍历层次结构时所耗用时间的差别。每种类型的创建时间确定使用缺省遍历顺序时在源和目标上的层次结构上所采用的顺序。确保每种类型的创建顺序在源层次结构和目标层次结构上是完全相同的，并且源和目标的结构完全相同。如果不能满足这些条件，那么选择用户指定的遍历顺序。

用户指定的遍历顺序

通过用户指定的遍历顺序，可在遍历顺序列表中定义要使用的相关类型。此顺序概述如何遍历层次结构及要导出的子表，而使用缺省遍历顺序则会导出层次结构中的所有表。

尽管在定义遍历顺序时确定了遍历层次结构的起始点及路径，但应记住，必须以*预定顺序*方式遍历子表。在启动新的分支之前，必须遍历层次结构中的每个分支直至底部。export 实用程序在指定的遍历顺序中查找是否存在违反此条件的情况。确保满足条件的一种方法是从层次结构顶部（或根表）开始，向下浏览层次结构（子表）直至底

部子表，然后返回至其超表，再向下浏览下一个“最右边的”子表，然后返回至更高级别的超表，再向下浏览至其子表，依此类推。

如果想要控制层次结构的遍历顺序，那么确保对 export 实用程序和 import 实用程序使用的是同一个遍历顺序。

示例 1

以下示例基于图 1 中的分层结构。要导出整个层次结构，请输入以下命令:

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO entire_hierarchy.ixf OF IXF HIERARCHY STARTING Person
```

注意，将参数 **HIERARCHY STARTING** 设置为 Person 指示缺省遍历顺序从表 PRESON 开始。

示例 2

要导出整个层次结构但仅导出年龄超过 20 岁的人员的数据，应输入以下命令:

```
DB2 CONNECT TO Source_db
   DB2 EXPORT TO entire_hierarchy.del OF DEL HIERARCHY (Person,
      Employee, Manager, Architect, Student) WHERE Age>=20
```

注意，将参数 **HIERARCHY** 设置为 Person, Employee, Manager, Architect, Student 将指示用户指定的遍历顺序。

标识列导出注意事项

可使用 export 实用程序以从包含标识列的表中导出数据。但是，标识列会限制您的输出文件格式选择。

如果对导出操作指定的 SELECT 语句的格式为 SELECT * FROM *tablename*，并且未使用 METHOD 选项，那么支持将标识列属性导出至 IXF 文件。然后可使用 IMPORT 命令的REPLACE_CREATE 和 CREATE 选项重新创建该表，包括其标识列属性。如果通过包含 GENERATED ALWAYS 类型标识列的表创建已导出 IXF 文件，那么成功导入数据文件的唯一方法就是在导入操作期间指定 `identityignore` 文件类型修饰符。否则会拒绝所有行（发出 SQL3550W）。

注: 不推荐使用 IMPORT 命令的 CREATE 和 REPLACE_CREATE 选项，将来的发行版中可能会除去这两个选项。

LOB 导出注意事项

导出包含大对象（LOB）列的表时，缺省操作是对每个 LOB 值导出最多 32 KB，以便将其与列数据的余下部分放在同一文件中。如果要导出超过 32 KB 的 LOB 值，那么应将 LOB 数据写至单独的文件以避免截断。

要指定应将 LOB 写至自己的文件，请使用 `lobsinfile` 文件类型修饰符。此修饰符指示 export 实用程序将 LOB 数据放在 LOBS TO 子句指定的目录中。使用 LOBS TO 或 LOBFILE 会隐式激活 `lobsinfile` 文件类型修饰符。缺省情况下，LOB 值与导出的关系数据将写至同一路径。如果使用 LOBS TO 选项指定了一个或多个路径，那么 export 实用程序将循环使用这些 LOB 路径，以便将每个成功的 LOB 值写入相应的 LOB 文件。还可使用 LOBFILE 选项对输出 LOB 文件指定名称。如果指定了 LOBFILE 选项，那么 lobfilename 的格式为 `lobfilespec.xxx.lob`，其中 `lobfilespec` 是为

LOBFILE 选项指定的值，而 xxx 是 export 实用程序生成的 lob 文件的序号。否则，lobfilename 的格式为: exportfilename.xxx.lob，其中 exportfilename 是为 EXPORT 命令指定的已导出输出文件的名称，而 xxx 是 export 实用程序生成的 lob 文件的序号。

缺省情况下，多个 LOB 将写至单个文件，但您也可指定将各个 LOB 存储在不同文件中。export 实用程序会生成 LOB 位置说明符（LLS），以允许将多个 LOB 存储在一个文件中。写至导出输出文件的 LLS 是一个指示 LOB 数据在文件中存储位置的字符串。LLS 的格式为 lobfilename.nnn.mmm/，其中 lobfilename.ext 是包含 LOB 的文件的名称，nnn 是该文件内 LOB 的偏移量（以字节为单位计量），而 mmm 是 LOB 的长度（以字节为单位计量）。例如，db2exp.001.123.456/ 的 LLS 表示 LOB 位于文件 db2exp.001 中，以 123 字节的偏移量开始进入文件，并且长度为 456 字节。如果 LLS 中指示的大小为 0，那么 LOB 被视为长度是 0。如果长度为 -1，那么 LOB 被视为 NULL，并且忽略偏移量和文件名。

如果不希望个别 LOB 数据并置于同一文件，请使用 lobsinsepfiles 文件类型修饰符以将每个 LOB 写至单独文件。

注: IXF 文件格式不会存储该列的 LOB 选项，如用于指示是否记录 LOB 列的选项。这意味着 import 实用程序不能重新创建包含定义为 1 GB 或更大的 LOB 列的表。

示例 1以下示例说明如何将 LOB（其中已导出 LOB 文件具有指定基本名称 lobs1）导出至 DEL 文件:

```
db2 export to myfile.del of del lobs to mylobs/
lobfile lobs1 modified by lobsinfile
select * from emp_photo
```

示例 2以下示例说明如何将 LOB 导出到 DEL 文件，其中每个 LOB 值将写至单独文件并且 lobfiles 将写至两个目录:

```
db2 export to myfile.del of del
      lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
select * from emp_photo
```

参考 - 导出

EXPORT

将数据从数据库中导出为几种外部文件格式之一。用户通过提供 SQL SELECT 语句或者类型表的分层信息来指定要导出的数据。

指向 第 20 页的『EXPORT 实用程序的文件类型修饰符』的快速链接。

权限

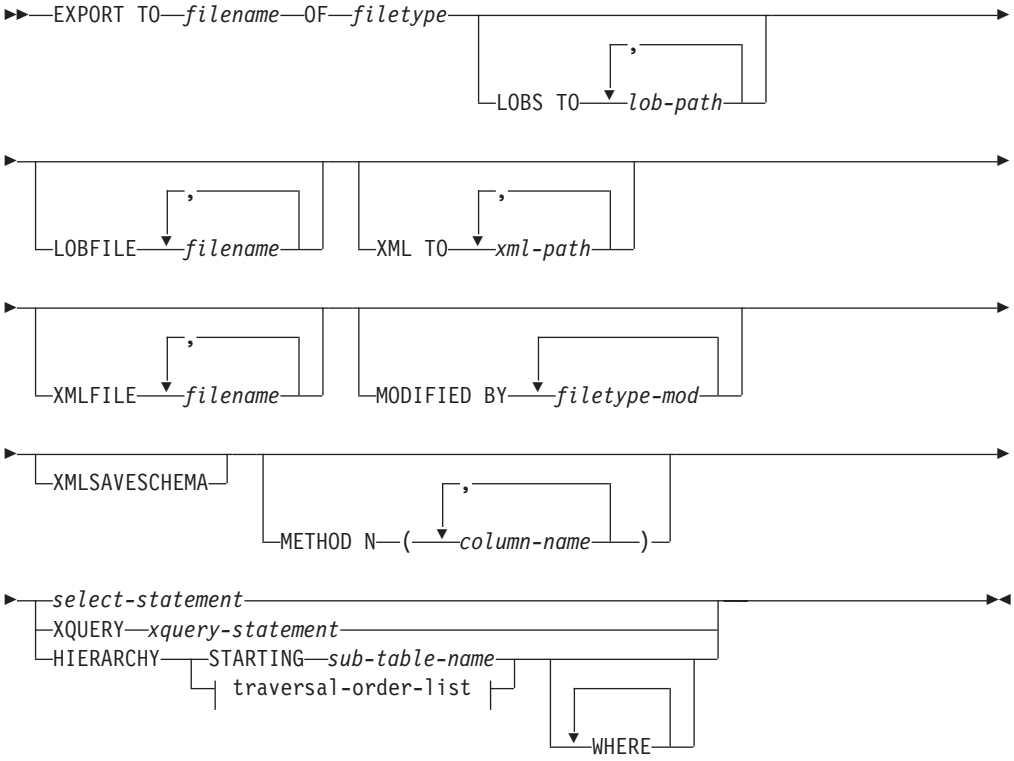
为下列其中一项:

- *sysadm*
- *dbadm*

或者对参与的每个表或视图的 CONTROL 或 SELECT 特权。

必需的连接

命令语法



traversal-order-list:

命令参数

HIERARCHY *traversal-order-list*

使用指定的遍历顺序导出子层次结构。必须按 PRE-ORDER 形式列示所有子表。第一个子表名将用作 SELECT 语句的目标表名。

HIERARCHY STARTING *sub-table-name*

使用缺省遍历顺序（对于 ASC、DEL 或 WSF 文件为 OUTER 顺序，或者是存储在 PC/IXF 数据文件中的顺序）导出从 *sub-table-name* 开始的子层次结构。

LOBFILE *filename*

为 LOB 文件指定一个或多个基本文件名。当耗尽了第一个名称的名称空间时，就使用第二个名称，以此类推。最多可指定 999 个文件名。这将隐式激活 LOBSINFILE 行为。

在导出操作期间创建 LOB 文件时，按以下方法来构造文件名：将此列表中的当前基本名称追加至（*lob-path* 中的）当前路径，然后再追加三位数的序号和三字符标识 lob。例如，如果当前 LOB 路径是 /u/foo/lob/path/ 目录，当前 LOB 文件名是 bar，那么创建的 LOB 文件将为 /u/foo/lob/path/bar.001.lob 和 /u/foo/lob/path/bar.002.lob 等等。

LOBS TO *lob-path*

指定要用来存储 LOB 文件的目录的一个或多个路径。每个 LOB 路径将至少有一个文件，而每个文件都将至少包含一个 LOB。最多可指定 999 个路径。这将隐式激活 LOBSINFILE 行为。

METHOD N *column-name*

指定要在输出文件中使用的一个或多个列名。如果未指定此参数，那么使用表中的列名。此参数仅对于 WSF 和 IXF 文件有效，但在导出分层数据时无效。

MODIFIED BY *filetype-mod*

指定文件类型修饰符选项。请参阅 第 20 页的『EXPORT 实用程序的文件类型修饰符』。

OF *filetype*

指定输出文件中数据的格式:

- DEL（定界 ASCII 格式），多种数据库管理器和文件管理器程序使用此格式。
- WSF（工作表格式），下列程序使用此格式:
  - Lotus® 1-2-3®
  - Lotus Symphony

  当导出 BIGINT 或 DECIMAL 数据时，只能正确导出属于类型为 DOUBLE 的范围内的值。尽管也会导出不属于此范围内的那些值，但是重新导入或装入这些值时可能会产生错误数据（取决于操作系统）。
- IXF（PC 版本的集成交换格式）是一种专有二进制格式。

select-statement

指定将返回要导出的数据的 SELECT 或 XQUERY 语句。如果该语句导致错误，那么会将消息写入消息文件（或标准输出）。如果错误代码为 SQL0012W、SQL0347W、SQL0360W、SQL0437W 或 SQL1824W，那么导出操作将继续执行；否则将停止。

TO *filename*

如果指定了已经存在的文件名，那么 EXPORT 实用程序将覆盖该文件的内容；它不会追加信息。

XMLFILE *filename*

为 XML 文件指定一个或多个基本文件名。当耗尽了第一个名称的名称空间时，就使用第二个名称，以此类推。

在导出操作期间创建 XML 文件时，按以下方法来构造文件名：将此列表中的当前基本名称追加至（*xml-path* 中的）当前路径，然后再追加三位数的序号和追加三字符标识 xml。例如，如果当前 XML 路径是 /u/foo/xml/path/ 目录，当前 XML 文件名是 bar，那么创建的 XML 文件将为 /u/foo/xml/path/bar.001.xml 和 /u/foo/xml/path/bar.002.xml 等等。

XML TO *xml-path*

指定要用来存储 XML 文件的目录的一个或多个路径。每个 XML 路径上将至少有一个文件，而每个文件都将至少包含一个 XQuery 数据模型（XDM）实例。如果指定了多个路径，那么 XDM 实例将平均分布在各个路径中。

XMLSAVESCHEMA

指定应保存所有 XML 列的 XML 模式信息。对于插入时要针对 XML 模式进

行验证的每个已导出的 XML 文档，该模式的标准 SQL 标识将作为一种（SCH）属性存储在相应的 XML 数据说明符（XDS）中。如果已导出的文档未针对 XML 模式进行验证，或者模式对象不再存在于数据库中，那么 SCH 属性将不会包含在相应的 XDS 中。

SQL 标识的模式和名称部分作为“OBJECTSCHEMA”和“OBJECTNAME”值存储在与 XML 模式相对应的 SYSCAT.XSROBJECTS 目录表的行中。

XMLSAVESCHEMA 选项与不能生成结构良好的 XML 文档的 XQuery 序列不兼容。

使用说明

- 在开始执行导出操作前，请确保完成所有表操作并释放所有锁定。这可以通过在关闭使用 WITH HOLD 打开的所有游标前发出 COMMIT，或者通过发出 ROLLBACK 来完成。
- 可以在 SELECT 语句中使用表别名。
- 放入消息文件中的消息包括从消息检索服务中返回的信息。每条消息都另起一行。
- 每当选择将长度超过 254 的字符列导出到 DEL 格式文件时，EXPORT 实用程序就会生成警告消息。
- 应该使用 PC/IXF 导入来在数据库之间移动数据。如果包含行分隔符的字符数据已导出到定界 ASCII（DEL）文件并且已由文本传输程序处理，那么包含行分隔符的字段将收缩或展开。
- 如果从同一台客户机可以访问源数据库和目标数据库，那么不需要执行文件复制步骤。
- 可以使用 DB2 Connect 从 DRDA® 服务器（例如，DB2 OS/390® 版、DB2 VM 和 VSE 版以及 DB2 OS/400® 版）中导出表。仅支持 PC/IXF 导出。
- 导出到 IXF 格式时，如果标识超出 IXF 格式支持的最大大小，那么导出会成功，但使用 CREATE 方式的后续导入操作不能使用生成的数据文件。将返回 SQL27984W。
- 如果在 Windows 上导出到软盘，并且单张软盘不足以容纳该表，那么系统将提示您插入另一张软盘，并且将生成多部件 PC/IXF 文件（又称为多卷 PC/IXF 文件或者逻辑上分割的 PC/IXF 文件）并将其存在单独的软盘上。在每个文件（最后一个文件除外）中，那么将写入一个 DB2 CONTINUATION RECORD（缩写为“AC”记录）以指示文件在逻辑上是分割的以及在何处查找下一个文件。随后可以将这些文件传送至 AIX® 系统，供 IMPORT 和 LOAD 实用程序读取。从 AIX 系统中调用 EXPORT 实用程序时，它将不会创建由多部分组成的 PC/IXF 文件。有关详细用法，请参阅 IMPORT 命令或 LOAD 命令。
- 如果所提供的 SELECT 语句采用 `SELECT * FROM tablename` 格式，那么 EXPORT 实用程序会将表的 NOT NULL WITH DEFAULT 属性存储在一个 IXF 文件中。
- 当导出类型表时，只能通过指定目标表名和 WHERE 子句来表示子查询语句。而在导出层次结构时，不能指定全查询和 *select-statement*。
- 对于不是 IXF 的文件格式，建议您指定遍历顺序列表，这是因为它会告诉 DB2 如何遍历层次结构以及要导出哪些子表。如果未指定此列表，那么将导出层次结构中的所有表，并且缺省顺序为 OUTER 顺序。还可以使用缺省顺序，也就是由 OUTER 函数给定的顺序。
- 在导入操作期间使用同样的遍历顺序。LOAD 实用程序不支持装入层次结构或子层次结构。

- 从具有受保护行的表中导出数据时，会话授权标识所拥有的 LBAC 凭证可能会限制导出的行。将不会导出会话授权标识对其不具有读访问权的那些行。不会提供任何错误或警告。
- 如果会话授权标识所拥有的 LBAC 凭证不允许读取导出操作中包括的一个或多个受保护列，那么导出操作会失败并返回错误（SQLSTATE 42512）。
- 由于导出程序包是使用 DATETIME ISO 格式绑定的，因此，当通过强制类型转换为字符串表示时，所有日期/时间/时间戳记值都将转换为 ISO 格式。由于 CLP 程序包是使用 DATETIME LOC 格式（特定于语言环境的格式）来绑定的，因此，如果 CLP DATETIME 格式不同于 ISO，那么您会发现 CLP 与 EXPORT 实用程序之间的行为不一致。例如，以下 SELECT 语句可能会返回预期的结果:

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
   COL2
   ----------
   05/10/2005
   05/10/2005
   05/10/2005
   3 record(s) selected.
```

但是使用同一 SELECT 子句的 EXPORT 命令将不会返回预期的结果:

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
   已导出的行数: 0
```

现在，将 LOCALE 日期格式替换为 ISO 格式将获得预期的结果:

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
   已导出的行数: 3
```

EXPORT 实用程序的文件类型修饰符

表 3. EXPORT 实用程序的有效文件类型修饰符: 所有文件格式

修饰符	描述
lobsinfile	*lob-path* 指定包含 LOB 数据的文件所在的路径。 每个路径都至少包含这样一个文件: 该文件至少包含数据文件中的“Lob 位置说明符”（LLS）所指向的一个 LOB。对于存储在 LOB 文件路径中的文件，LLS 就是对这些文件中的 LOB 所在位置的字符串表示。LLS 的格式为 *filename.ext.nnn.mmm/*，其中 *filename.ext* 是包含 LOB 的文件名，*nnn* 是文件中的 LOB 的偏移量（以字节计），*mmm* 是 LOB 的长度（以字节计）。例如，如果 db2exp.001.123.456/ 字符串存储在数据文件中，那么 LOB 位于 db2exp.001 文件中偏移量为 123 的位置，其长度为 456 字节。 如果在使用 EXPORT 实用程序时指定“lobsinfile”修饰符，那么会将 LOB 数据放置在 LOBS TO 子句所指定的位置。否则，会将 LOB 数据发送至数据文件目录。LOBS TO 子句指定要用来存储 LOB 文件的目录的一个或多个路径。每个 LOB 路径将至少有一个文件，而每个文件都将至少包含一个 LOB。LOBS TO 或 LOBFILE 选项将隐式激活 LOBSINFILE 行为。 要指示一个空 LOB，应将大小输入为 -1。如果将大小指定为 0，那么会将它视作长度为 0 的 LOB。对于长度为 -1 的 LOBS，将忽略偏移量和文件名。例如，空 LOB 的 LLS 可能是 db2exp.001.7.-1/。
xmlinsepfiles	将每个 XQuery 数据模型（XDM）实例都写入一个单独的文件中。缺省情况下，多个值同时并置在同一文件中。

表 3. EXPORT 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
lobsinsepfiles	每个 LOB 值都将写入一个单独的文件中。缺省情况下，多个值同时并置在同一文件中。
xmlnodeclaration	写入 XDM 实例时不带 XML 声明标记。缺省情况下，导出 XDM 实例时都会在其开头带有包含编码属性的 XML 声明标记。
xmlchar	采用字符代码页来编写 XDM 实例。注意，字符代码页就是由文件类型修饰符 `codepage` 指定的值；如果未指定该修饰符，那么字符代码页就是应用程序代码页。缺省情况下，采用 Unicode 来写出 XDM 实例。
xmlgraphic	如果对 EXPORT 命令指定了 `xmlgraphic` 修饰符，那么无论应用程序代码页或 `codepage` 文件类型修饰符如何，都将采用 UTF-16 代码页来对已导出的 XML 文档进行编码。

表 4. EXPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式

修饰符	描述

chardel*x*	*x* 是单个字符串定界符。缺省值是双引号（"）。使用指定的字符而不是使用双引号将字符串引起来[2]。如果您想显式地指定双引号作为字符串定界符，那么应按如下所示指定双引号:
	`modified by chardel""`
	也可以指定单引号（'）作为字符串定界符，如下所示:
	`modified by chardel''`

codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输出数据集中的数据的代码页。在导出操作期间，将采用应用程序代码页的字符数据转换为采用此代码页。 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。`codepage` 修饰符不能与 `lobsinfile` 修饰符一起使用。

coldel*x*	*x* 是一个单字符列定界符。缺省值是逗号（,）。使用指定字符而不是逗号来表示列的末尾[2]。
	在以下示例中，`coldel;` 会导致 EXPORT 实用程序将分号字符（;）用作已导出的数据的列定界符:
	`db2 "export to temp of del modified by coldel;`
	`select * from staff where dept = 20"`

decplusblank	加号字符。导致在正的十进制值前面加上空格而不是加号（+）。缺省操作是在正的十进制值前面加上加号。
decpt*x*	*x* 是单个字符，它取代句点作为小数点字符。缺省值是句点（.）。使用指定字符而不是句点作为小数点字符[2]。
nochardel	在列数据两边将不会添加字符定界符。如果打算使用 DB2 来导入或装入数据，那么不应指定此选项。提供此修饰符的目的是支持不具有字符定界符的供应商数据文件。未正确使用此修饰符可能会导致数据丢失或毁坏。 不能将此选项与 `chardelx` 或 `nodoubledel` 同时指定。它们是互斥选项。
nodoubledel	不识别双字符定界符[2]。

表 4. EXPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式 (续)

修饰符	描述

striplzeros	从导出的所有十进制列中除去前导零。
	
	请考虑以下示例:
	
	`db2 create table decimalTable ( c1 decimal( 31, 2 ) )`
	`db2 insert into decimalTable values ( 1.1 )`
	
	`db2 export to data of del select * from decimalTable`
	
	`db2 export to data of del modified by STRIPLZEROS`
	`select * from decimalTable`
	
	在第一个导出操作中，导出的文件数据的内容将为 +0000000000000000000000000000001.10。在第二个操作中（除了 striplzeros 修饰符之外，它与第一个操作完全相同），导出的文件数据的内容将为 +1.10。

表 4. EXPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式 (续)

修饰符	描述

timestampformat="*x*"	*x* 是源文件中的时间戳记格式[4]。有效时间戳记元素包括:
	
	YYYY - 年份（四位数，范围是 0000 到 9999）
	M - 月份（一位数或两位数，范围是 1 到 12）
	MM - 月份（两位数，范围是 01 到 12；与 M 和 MMM 元素互斥）
	MMM - 月份（由三个不区分大小写的字母组成的月份名称缩写；与 M 和 MM 元素互斥）
	D - 日（一位数或两位数，范围是 1 到 31）
	DD - 日（两位数，范围是 1 到 31；与 D 元素互斥）
	DDD - 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥）
	H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24）
	HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥）
	M - 分钟（一位数或两位数，范围是 0 到 59）
	MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥）
	S - 秒（一位数或两位数，范围是 0 到 59）
	SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥）
	SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥）
	UUUUUU - 微秒（六位数，范围是 000000 到 999999；与所有其他微秒元素互斥）
	UUUUU - 微秒（五位数，范围是 00000 到 99999，映射至范围 000000 到 999990；与所有其他微秒元素互斥）
	UUUU - 微秒（四位数，范围是 0000 到 9999，映射至范围 000000 到 999900；与所有其他微秒元素互斥）
	UUU - 微秒（三位数，范围是 000 到 999，映射至范围 000000 到 999000；与所有其他微秒元素互斥）
	UU - 微秒（两位数，范围是 00 到 99，映射至范围 000000 到 990000；与所有其他微秒元素互斥）
	U - 微秒（一位数，范围是 0 到 9，映射至范围 000000 到 900000；与所有其他微秒元素互斥）
	TT - 正午指示符（AM 或 PM）
	
	以下是一个表示时间戳记格式的示例:
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 元素将生成下列值: 'Jan'、'Feb'、'Mar'、'Apr'、'May'、'Jun'、'Jul'、'Aug'、'Sep'、'Oct'、'Nov' 和 'Dec'。'Jan' 表示 1 月份，'Dec' 表示 12 月份。
	
	以下示例说明如何从称为“schedule”的表中导出包含用户定义的时间戳记格式的数据:
	
	`db2 export to delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`select * from schedule`

表 5. EXPORT 实用程序的有效文件类型修饰符: IXF 文件格式

修饰符	描述
codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输出数据集中的数据的代码页。在导出操作期间，将采用此代码页的字符数据转换为采用应用程序代码页。 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。codepage 修饰符不能与 lobsinfile 修饰符一起使用。

表 6. EXPORT 实用程序的有效文件类型修饰符: WSF 文件格式

修饰符	描述
1	创建与 Lotus 1-2-3 发行版 1 或 Lotus 1-2-3 发行版 1a 兼容的 WSF 文件[5]。这是缺省情况。
2	创建与 Lotus Symphony 发行版 1.0 兼容的 WSF 文件[5]。
3	创建与 Lotus 1-2-3 版本 2 或 Lotus Symphony 发行版 1.1 兼容的 WSF 文件[5]。
4	创建包含 DBCS 字符的 WSF 文件。

注:

1. 如果您尝试将不受支持的文件类型与 MODIFIED BY 选项配合使用，那么 EXPORT 实用程序将不会发出警告。如果尝试这样做，导出操作将失败，并且会返回错误代码。
2. *移动数据时的定界符注意事项*列示了可以用作定界符的字符存在的限制。
3. EXPORT 实用程序通常:
   - 采用 *YYYYMMDD* 格式写入日期数据
   - 采用 *"YYYY-MM-DD"* 格式写入 char(date) 数据
   - 采用 *"HH.MM.SS"* 格式写入时间数据
   - 采用 *"YYYY-MM-DD-HH.MM.SS.uuuuuu"* 格式写入时间戳记数据

   在导出操作的 SELECT 语句中指定的任何日期时间列中包含的数据也将采用这些格式。
4. 对于时间戳记格式，必须要注意避免月份描述符与分钟描述符之间的不明确性，这是因为它们都使用字母 M。月份字段必须与其他日期字段相邻。而分钟字段必须与其他时间字段相邻。以下是一些不明确的时间戳记格式:

   ```
   "M"（既可能是月份，也可能是分钟）
   "M:M"（无法区分哪个是月份，哪个是分钟）
   "M:YYYY:M"（两者都将被解释为月份。）
   "S:M:YYYY"（与时间值和日期值都相邻）
   ```

   在不明确的情况下，实用程序将报告一条错误消息，并且操作将失败。

   以下是一些明确的时间戳记格式:

   ```
   "M:YYYY"（表示月份）
   "S:M"（表示分钟）
   "M:YYYY:S:M"（前者表示月份，后者表示分钟）
   "M:H:YYYY:M:D"（前者表示分钟，后者表示月份）
   ```
5. 还可以通过在参数字符串 *filetype-mod* 中指定 L 来表示 Lotus 1-2-3，或者指定 S 来表示 Symphony，来将这些文件定向至特定产品。只能指定一个值或产品指示符。
6. XML 列不支持 WSF 文件格式。
7. 即使既未指定 XMLFILE 子句也未指定 XML TO 子句，也会将所有 XDM 实例写入与主数据文件分开的 XML 文件中。缺省情况下，XML 文件将写入已导出数据文件的路径中。XML 文件的缺省基本名称是已导出数据文件的名称以及对它追加的扩展名".xml"。
8. 除非指定了 XMLNODECLARATION 文件类型修饰符，否则，写入所有 XDM 实例时都会在其开头带有包含编码属性的 XML 声明。
9. 缺省情况下，除非指定了 XMLCHAR 或 XMLGRAPHIC 文件类型修饰符，否则，所有 XDM 实例都采用 Unicode 编写。

10. XML 数据和 LOB 数据的缺省路径就是主数据文件的路径。缺省 XML 文件基本名称是主数据文件。缺省 LOB 文件基本名称是主数据文件。例如，如果主数据文件是

    ```
    /mypath/myfile.del
    ```

    那么 XML 数据和 LOB 数据的缺省路径是

    ```
    /mypath
    ```

    缺省 XML 文件基本名称是

    ```
    myfile.del
    ```

    而缺省 LOB 文件基本名称是

    ```
    myfile.del
    ```

    .

    必须指定文件类型修饰符 LOBSINFILE，才能生成 LOB 文件。
11. EXPORT 实用程序将对每个 LOB 文件或 XML 文件追加一个数字标识。该标识是一个至少具有 3 位数的序列值（位数不足时填充 0），起始值为

    ```
    .001
    ```

    在第 999 个 LOB 文件或 XML 文件之后，将不再为该标识填充零（例如，第 1000 个 LOG 文件或 XML 文件的扩展名将为

    ```
    .1000
    ```

    紧接着该数字标识后面是一个用来表示数据类型的三字符类型标识（

    ```
    .lob
    ```

    或

    ```
    .xml
    ```

    ）例如，生成的 LOB 文件的名称格式为

    ```
    myfile.del.001.lob
    ```

    ，而生成的 XML 文件的名称格式为

    ```
    myfile.del.001.xml
    ```
12. 对于是一些格式不正确的文档的 XDM 实例，可以让 EXPORT 实用程序通过指定 XQuery 来将其导出。但是，不能直接将这些导出的文档导入或装入 XML 列中，因为 XML 列只能包含完整的文档。

使用 ADMIN_CMD 过程的 EXPORT 命令

将数据从数据库中导出为几种外部文件格式之一。用户通过提供 SQL SELECT 语句或者类型表的分层信息来指定要导出的数据。

指向 第 30 页的『EXPORT 实用程序的文件类型修饰符』的快速链接。

权限

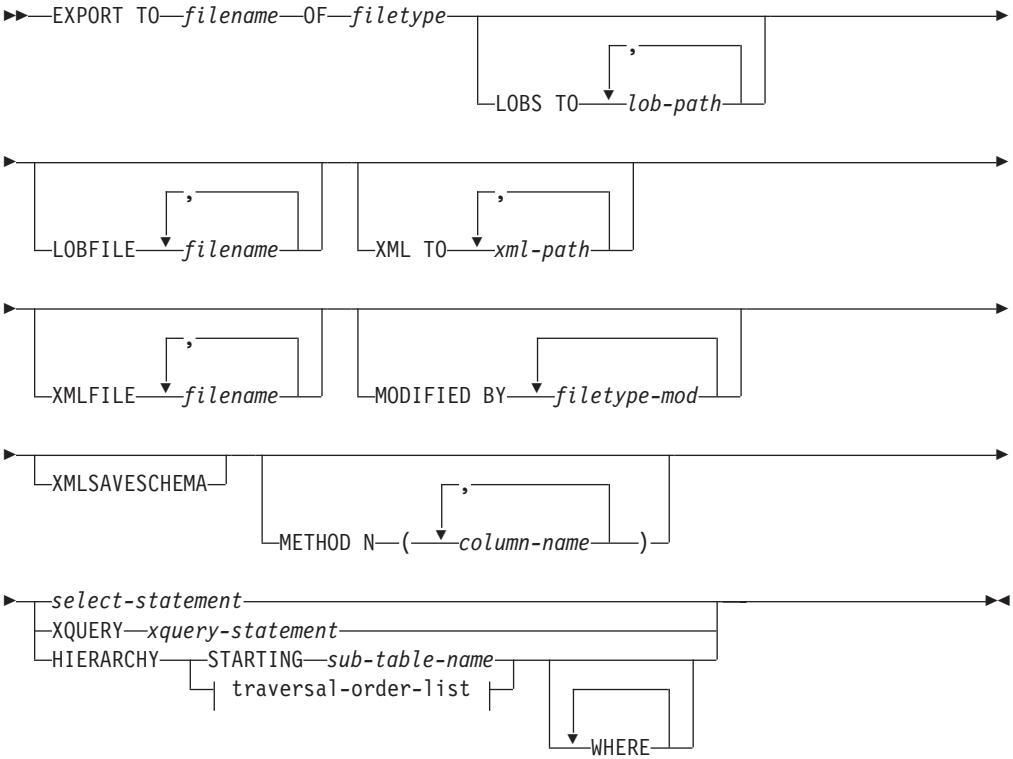
为下列其中一项:

- *sysadm*
- *dbadm*

或者对参与的每个表或视图的 CONTROL 或 SELECT 特权。

必需的连接

命令语法



traversal-order-list:

命令参数

HIERARCHY *traversal-order-list*

使用指定的遍历顺序导出子层次结构。必须按 PRE-ORDER 形式列示所有子表。第一个子表名将用作 SELECT 语句的目标表名。

HIERARCHY STARTING *sub-table-name*

使用缺省遍历顺序（对于 ASC、DEL 或 WSF 文件为 OUTER 顺序，或者是存储在 PC/IXF 数据文件中的顺序）导出从 *sub-table-name* 开始的子层次结构。

LOBFILE *filename*

为 LOB 文件指定一个或多个基本文件名。当耗尽了第一个名称的名称空间时，就使用第二个名称，以此类推。最多可指定 999 个文件名。这将隐式激活 LOBSINFILE 行为。

在导出操作期间创建 LOB 文件时，按以下方法来构造文件名：将此列表中的当前基本名称追加至（*lob-path* 中的）当前路径，然后再追加三位数的序号和三字符标识 lob。例如，如果当前 LOB 路径是 /u/foo/lob/path/ 目录，当前 LOB 文件名是 bar，那么创建的 LOB 文件将为 /u/foo/lob/path/bar.001.lob 和 /u/foo/lob/path/bar.002.lob 等等。

LOBS TO *lob-path*

指定要用来存储 LOB 文件的目录的一个或多个路径。每个 LOB 路径将至少有一个文件，而每个文件都将至少包含一个 LOB。最多可指定 999 个路径。这将隐式激活 LOBSINFILE 行为。

METHOD N *column-name*

指定要在输出文件中使用的一个或多个列名。如果未指定此参数，那么使用表中的列名。此参数仅对于 WSF 和 IXF 文件有效，但在导出分层数据时无效。

MODIFIED BY *filetype-mod*

指定文件类型修饰符选项。请参阅 第 30 页的『EXPORT 实用程序的文件类型修饰符』。

OF *filetype*

指定输出文件中数据的格式：

- DEL（定界 ASCII 格式），多种数据库管理器和文件管理器程序使用此格式。
- WSF（工作表格式），下列程序使用此格式：
  - Lotus 1-2-3
  - Lotus Symphony

  当导出 BIGINT 或 DECIMAL 数据时，只能正确导出属于类型为 DOUBLE 的范围内的值。尽管也会导出不属于此范围内的那些值，但是重新导入或装入这些值时可能会产生错误数据（取决于操作系统）。
- IXF（PC 版本的集成交换格式）是一种专有二进制格式。

select-statement

指定将返回要导出的数据的 SELECT 或 XQUERY 语句。如果该语句导致错误，那么会将消息写入消息文件（或标准输出）。如果错误代码为 SQL0012W、SQL0347W、SQL0360W、SQL0437W 或 SQL1824W，那么导出操作将继续执行；否则将停止。

TO *filename*

如果指定了已经存在的文件名，那么 EXPORT 实用程序将覆盖该文件的内容；它不会追加信息。

XMLFILE *filename*

为 XML 文件指定一个或多个基本文件名。当耗尽了第一个名称的名称空间时，就使用第二个名称，以此类推。

在导出操作期间创建 XML 文件时，按以下方法来构造文件名：将此列表中的当前基本名称追加至（*xml-path* 中的）当前路径，然后再追加三位数的序号和追

加三字符标识 xml。例如，如果当前 XML 路径是 /u/foo/xml/path/ 目录，当前 XML 文件名是 bar，那么创建的 XML 文件将为 /u/foo/xml/path/bar.001.xml 和 /u/foo/xml/path/bar.002.xml 等等。

XML TO *xml-path*

指定要用来存储 XML 文件的目录的一个或多个路径。每个 XML 路径上将至少有一个文件，而每个文件都将至少包含一个 XQuery 数据模型（XDM）实例。如果指定了多个路径，那么 XDM 实例将平均分布在各个路径中。

XMLSAVESCHEMA

指定应保存所有 XML 列的 XML 模式信息。对于插入时要针对 XML 模式进行验证的每个已导出的 XML 文档，该模式的标准 SQL 标识将作为一种（SCH）属性存储在相应的 XML 数据说明符（XDS）中。如果已导出的文档未针对 XML 模式进行验证，或者模式对象不再存在于数据库中，那么 SCH 属性将不会包含在相应的 XDS 中。

SQL 标识的模式和名称部分作为“OBJECTSCHEMA”和“OBJECTNAME”值存储在与 XML 模式相对应的 SYSCAT.XSROBJECTS 目录表的行中。

XMLSAVESCHEMA 选项与不能生成结构良好的 XML 文档的 XQuery 序列不兼容。

使用说明

- 在开始执行导出操作前，请确保完成所有表操作并释放所有锁定。这可以通过在关闭使用 WITH HOLD 打开的所有游标前发出 COMMIT，或者通过发出 ROLLBACK 来完成。
- 可以在 SELECT 语句中使用表别名。
- 放入消息文件中的消息包括从消息检索服务中返回的信息。每条消息都另起一行。
- 每当选择将长度超过 254 的字符列导出到 DEL 格式文件时，EXPORT 实用程序就会生成警告消息。
- 应该使用 PC/IXF 导入来在数据库之间移动数据。如果包含行分隔符的字符数据已导出到定界 ASCII（DEL）文件并且已由文本传输程序处理，那么包含行分隔符的字段将收缩或展开。
- 如果从同一台客户机可以访问源数据库和目标数据库，那么不需要执行文件复制步骤。
- 可以使用 DB2 Connect 从 DRDA 服务器（例如，DB2 OS/390 版、DB2 VM 和 VSE 版以及 DB2 OS/400 版）中导出表。仅支持 PC/IXF 导出。
- 导出到 IXF 格式时，如果标识超出 IXF 格式支持的最大大小，那么导出会成功，但使用 CREATE 方式的后续导入操作不能使用生成的数据文件。将返回 SQL27984W。
- 如果在 Windows 上导出到软盘，并且单张软盘不足以容纳该表，那么系统将提示您插入另一张软盘，并且将生成多部件 PC/IXF 文件（又称为多卷 PC/IXF 文件或者逻辑上分割的 PC/IXF 文件）并将其存在单独的软盘上。在每个文件（最后一个文件除外）中，那么将写入一个 DB2 CONTINUATION RECORD（缩写为“AC”记录）以指示文件在逻辑上是分割的以及在何处查找下一个文件。随后可以将这些文件传送至 AIX 系统，供 IMPORT 和 LOAD 实用程序读取。从 AIX 系统中调用 EXPORT 实用程序时，它将不会创建由多部分组成的 PC/IXF 文件。有关详细用法，请参阅 IMPORT 命令或 LOAD 命令。
- 如果所提供的 SELECT 语句采用 SELECT * FROM tablename 格式，那么 EXPORT 实用程序会将表的 NOT NULL WITH DEFAULT 属性存储在一个 IXF 文件中。

- 当导出类型表时，只能通过指定目标表名和 WHERE 子句来表示子查询语句。而在导出层次结构时，不能指定全查询和 *select-statement*。
- 对于不是 IXF 的文件格式，建议您指定遍历顺序列表，这是因为它会告诉 DB2 如何遍历层次结构以及要导出哪些子表。如果未指定此列表，那么将导出层次结构中的所有表，并且缺省顺序为 OUTER 顺序。还可以使用缺省顺序，也就是由 OUTER 函数给定的顺序。
- 在导入操作期间使用同样的遍历顺序。LOAD 实用程序不支持装入层次结构或子层次结构。
- 从具有受保护行的表中导出数据时，会话授权标识所拥有的 LBAC 凭证可能会限制导出的行。将不会导出会话授权标识对其不具有读访问权的那些行。不会提供任何错误或警告。
- 如果会话授权标识所拥有的 LBAC 凭证不允许读取导出操作中包括的一个或多个受保护列，那么导出操作会失败并返回错误（SQLSTATE 42512）。
- 由于导出程序包是使用 DATETIME ISO 格式绑定的，因此，当通过强制类型转换为字符串表示时，所有日期/时间/时间戳记值都将转换为 ISO 格式。由于 CLP 程序包是使用 DATETIME LOC 格式（特定于语言环境的格式）来绑定的，因此，如果 CLP DATETIME 格式不同于 ISO，那么您会发现 CLP 与 EXPORT 实用程序之间的行为不一致。例如，以下 SELECT 语句可能会返回预期的结果:

  ```
  db2 select col2 from tab1 where char(col2)='05/10/2005';
     COL2
     ----------
     05/10/2005
     05/10/2005
     05/10/2005
     3 record(s) selected.
  ```

  但是使用同一 SELECT 子句的 EXPORT 命令将不会返回预期的结果:

  ```
  db2 export to test.del of del select col2 from test
  where char(col2)='05/10/2005';
     已导出的行数: 0
  ```

  现在，将 LOCALE 日期格式替换为 ISO 格式将获得预期的结果:

  ```
  db2 export to test.del of del select col2 from test
  where char(col2)='2005-05-10';
     已导出的行数: 3
  ```

EXPORT 实用程序的文件类型修饰符

表 7. EXPORT 实用程序的有效文件类型修饰符: 所有文件格式

修饰符	描述
lobsinfile	*lob-path* 指定包含 LOB 数据的文件所在的路径。 每个路径都至少包含这样一个文件: 该文件至少包含数据文件中的“Lob 位置说明符”（LLS）所指向的一个 LOB。对于存储在 LOB 文件路径中的文件，LLS 就是对这些文件中的 LOB 所在位置的字符串表示。LLS 的格式为 *filename.ext.nnn.mmm/*，其中 *filename.ext* 是包含 LOB 的文件名，*nnn* 是文件中的 LOB 的偏移量（以字节计），*mmm* 是 LOB 的长度（以字节计）。例如，如果 `db2exp.001.123.456/` 字符串存储在数据文件中，那么 LOB 位于 `db2exp.001` 文件中偏移量为 123 的位置，其长度为 456 字节。 如果在使用 EXPORT 实用程序时指定“lobsinfile”修饰符，那么会将 LOB 数据放置在 LOBS TO 子句所指定的位置。否则，会将 LOB 数据发送至数据文件目录。LOBS TO 子句指定要用来存储 LOB 文件的目录的一个或多个路径。每个 LOB 路径将至少有一个文件，而每个文件都将至少包含一个 LOB。LOBS TO 或 LOBFILE 选项将隐式激活 LOBSINFILE 行为。 要指示一个空 LOB，应将大小输入为 -1。如果将大小指定为 0，那么会将它视作长度为 0 的 LOB。对于长度为 -1 的 LOBS，将忽略偏移量和文件名。例如，空 LOB 的 LLS 可能是 db2exp.001.7.-1/。
xmlinsepfiles	将每个 XQuery 数据模型（XDM）实例都写入一个单独的文件中。缺省情况下，多个值同时并置在同一文件中。
lobsinsepfiles	每个 LOB 值都将写入一个单独的文件中。缺省情况下，多个值同时并置在同一文件中。
xmlnodeclaration	写入 XDM 实例时不带 XML 声明标记。缺省情况下，导出 XDM 实例时都会在其开头带有包含编码属性的 XML 声明标记。
xmlchar	采用字符代码页来编写 XDM 实例。注意，字符代码页就是由文件类型修饰符 `codepage` 指定的值；如果未指定该修饰符，那么字符代码页就是应用程序代码页。缺省情况下，采用 Unicode 来写出 XDM 实例。
xmlgraphic	如果对 EXPORT 命令指定了 `xmlgraphic` 修饰符，那么无论应用程序代码页或 `codepage` 文件类型修饰符如何，都将采用 UTF-16 代码页来对已导出的 XML 文档进行编码。

表 8. EXPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式

修饰符	描述

chardel*x*	*x* 是单个字符串定界符。缺省值是双引号（″）。使用指定的字符而不是使用双引号将字符串引起来[2]。如果您想显式地指定双引号作为字符串定界符，那么应按如下所示指定双引号:
	
	`modified by chardel""`
	
	也可以指定单引号（'）作为字符串定界符，如下所示:
	
	`modified by chardel''`

codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输出数据集中的数据的代码页。在导出操作期间，将采用应用程序代码页的字符数据转换为采用此代码页。 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。`codepage` 修饰符不能与 `lobsinfile` 修饰符一起使用。

表 8. EXPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式 (续)

修饰符	描述

coldel*x*	*x* 是一个单字符列定界符。缺省值是逗号（,）。使用指定字符而不是逗号来表示列的末尾[2]。
	
	在以下示例中，`coldel;` 会导致 EXPORT 实用程序将分号字符（;）用作已导出的数据的列定界符:
	
	`db2 "export to temp of del modified by coldel;`
	`select * from staff where dept = 20"`

decplusblank	加号字符。导致在正的十进制值前面加上空格而不是加号（+）。缺省操作是在正的十进制值前面加上加号。
decpt*x*	*x* 是单个字符，它取代句点作为小数点字符。缺省值是句点（.）。使用指定字符而不是句点作为小数点字符[2]。
nochardel	在列数据两边将不会添加字符定界符。如果打算使用 DB2 来导入或装入数据，那么不应指定此选项。提供此修饰符的目的是支持不具有字符定界符的供应商数据文件。未正确使用此修饰符可能会导致数据丢失或毁坏。 不能将此选项与 `chardelx` 或 `nodoubledel` 同时指定。它们是互斥选项。
nodoubledel	不识别双字符定界符[2]。

striplzeros	从导出的所有十进制列中除去前导零。
	
	请考虑以下示例:
	
	`db2 create table decimalTable ( c1 decimal( 31, 2 ) )`
	`db2 insert into decimalTable values ( 1.1 )`
	
	`db2 export to data of del select * from decimalTable`
	
	`db2 export to data of del modified by STRIPLZEROS`
	`select * from decimalTable`
	
	在第一个导出操作中，导出的文件数据的内容将为 +0000000000000000000000000000001.10。在第二个操作中（除了 `striplzeros` 修饰符之外，它与第一个操作完全相同），导出的文件数据的内容将为 +1.10。

表 8. EXPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式 (续)

修饰符	描述

timestampformat="*x*"	*x* 是源文件中的时间戳记格式[4]。有效时间戳记元素包括:
	
	YYYY - 年份（四位数，范围是 0000 到 9999）
	M - 月份（一位数或两位数，范围是 1 到 12）
	MM - 月份（两位数，范围是 01 到 12；与 M 和 MMM 元素互斥）
	MMM - 月份（由三个不区分大小写的字母组成的月份名称缩写；与 M 和 MM 元素互斥）
	D - 日（一位数或两位数，范围是 1 到 31）
	DD - 日（两位数，范围是 1 到 31；与 D 元素互斥）
	DDD - 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥）
	H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24）
	HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥）
	M - 分钟（一位数或两位数，范围是 0 到 59）
	MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥）
	S - 秒（一位数或两位数，范围是 0 到 59）
	SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥）
	SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥）
	UUUUUU - 微秒（六位数，范围是 000000 到 999999；与所有其他微秒元素互斥）
	UUUUU - 微秒（五位数，范围是 00000 到 99999，映射至范围 000000 到 999990；与所有其他微秒元素互斥）
	UUUU - 微秒（四位数，范围是 0000 到 9999，映射至范围 000000 到 999900；与所有其他微秒元素互斥）
	UUU - 微秒（三位数，范围是 000 到 999，映射至范围 000000 到 999000；与所有其他微秒元素互斥）
	UU - 微秒（两位数，范围是 00 到 99，映射至范围 000000 到 990000；与所有其他微秒元素互斥）
	U - 微秒（一位数，范围是 0 到 9，映射至范围 000000 到 900000；与所有其他微秒元素互斥）
	TT - 正午指示符（AM 或 PM）
	
	以下是一个表示时间戳记格式的示例:
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 元素将生成下列值: 'Jan'、'Feb'、'Mar'、'Apr'、'May'、'Jun'、'Jul'、'Aug'、'Sep'、'Oct'、'Nov' 和 'Dec'。'Jan' 表示 1 月份，'Dec' 表示 12 月份。
	
	以下示例说明如何从称为“schedule”的表中导出包含用户定义的时间戳记格式的数据:
	
	`db2 export to delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`select * from schedule`

表 9. EXPORT 实用程序的有效文件类型修饰符: IXF 文件格式

修饰符	描述
codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输出数据集中的数据的代码页。在导出操作期间，将采用此代码页的字符数据转换为采用应用程序代码页。 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。codepage 修饰符不能与 lobsinfile 修饰符一起使用。

表 10. EXPORT 实用程序的有效文件类型修饰符: WSF 文件格式

修饰符	描述
1	创建与 Lotus 1-2-3 发行版 1 或 Lotus 1-2-3 发行版 1a 兼容的 WSF 文件[5]。这是缺省情况。
2	创建与 Lotus Symphony 发行版 1.0 兼容的 WSF 文件[5]。
3	创建与 Lotus 1-2-3 版本 2 或 Lotus Symphony 发行版 1.1 兼容的 WSF 文件[5]。
4	创建包含 DBCS 字符的 WSF 文件。

注:

1. 如果您尝试将不受支持的文件类型与 MODIFIED BY 选项配合使用，那么 EXPORT 实用程序将不会发出警告。如果尝试这样做，导出操作将失败，并且会返回错误代码。
2. *移动数据时的定界符注意事项*列示了可以用作定界符的字符存在的限制。
3. EXPORT 实用程序通常:
   - 采用 *YYYYMMDD* 格式写入日期数据
   - 采用 *"YYYY-MM-DD"* 格式写入 char(date) 数据
   - 采用 *"HH.MM.SS"* 格式写入时间数据
   - 采用 *"YYYY-MM-DD-HH.MM.SS.uuuuuu"* 格式写入时间戳记数据

   在导出操作的 SELECT 语句中指定的任何日期时间列中包含的数据也将采用这些格式。
4. 对于时间戳记格式，必须要注意避免月份描述符与分钟描述符之间的不明确性，这是因为它们都使用字母 M。月份字段必须与其他日期字段相邻。而分钟字段必须与其他时间字段相邻。以下是一些不明确的时间戳记格式:

   ```
   "M"（既可能是月份，也可能是分钟）
   "M:M"（无法区分哪个是月份，哪个是分钟）
   "M:YYYY:M"（两者都将被解释为月份。）
   "S:M:YYYY"（与时间值和日期值都相邻）
   ```

   在不明确的情况下，实用程序将报告一条错误消息，并且操作将失败。

   以下是一些明确的时间戳记格式:

   ```
   "M:YYYY"（表示月份）
   "S:M"（表示分钟）
   "M:YYYY:S:M"（前者表示月份，后者表示分钟）
   "M:H:YYYY:M:D"（前者表示分钟，后者表示月份）
   ```
5. 还可以通过在参数字符串 *filetype-mod* 中指定 L 来表示 Lotus 1-2-3，或者指定 S 来表示 Symphony，来将这些文件定向至特定产品。只能指定一个值或产品指示符。
6. XML 列不支持 WSF 文件格式。
7. 即使既未指定 XMLFILE 子句也未指定 XML TO 子句，也会将所有 XDM 实例写入与主数据文件分开的 XML 文件中。缺省情况下，XML 文件将写入已导出数据文件的路径中。XML 文件的缺省基本名称是已导出数据文件的名称以及对它追加的扩展名".xml"。
8. 除非指定了 XMLNODECLARATION 文件类型修饰符，否则，写入所有 XDM 实例时都会在其开头带有包含编码属性的 XML 声明。
9. 缺省情况下，除非指定了 XMLCHAR 或 XMLGRAPHIC 文件类型修饰符，否则，所有 XDM 实例都采用 Unicode 编写。

10. XML 数据和 LOB 数据的缺省路径就是主数据文件的路径。缺省 XML 文件基本名称是主数据文件。缺省 LOB 文件基本名称是主数据文件。例如，如果主数据文件是

    ```
    /mypath/myfile.del
    ```

    那么 XML 数据和 LOB 数据的缺省路径是

    ```
    /mypath
    ```

    缺省 XML 文件基本名称是

    ```
    myfile.del
    ```

    而缺省 LOB 文件基本名称是

    ```
    myfile.del
    ```

    .

    必须指定文件类型修饰符 LOBSINFILE，才能生成 LOB 文件。
11. EXPORT 实用程序将对每个 LOB 文件或 XML 文件追加一个数字标识。该标识是一个至少具有 3 位数的序列值（位数不足时填充 0），起始值为

    ```
    .001
    ```

    在第 999 个 LOB 文件或 XML 文件之后，将不再为该标识填充零（例如，第 1000 个 LOG 文件或 XML 文件的扩展名将为

    ```
    .1000
    ```

    紧接着该数字标识后面是一个用来表示数据类型的三字符类型标识（

    ```
    .lob
    ```

    或

    ```
    .xml
    ```

    ）例如，生成的 LOB 文件的名称格式为

    ```
    myfile.del.001.lob
    ```

    ，而生成的 XML 文件的名称格式为

    ```
    myfile.del.001.xml
    ```
12. 对于是一些格式不正确的文档的 XDM 实例，可以让 EXPORT 实用程序通过指定 XQuery 来将其导出。但是，不能直接将这些导出的文档导入或装入 XML 列中，因为 XML 列只能包含完整的文档。

db2Export - 从数据库中导出数据

将数据从数据库中导出为几种外部文件格式之一。用户通过提供 SQL SELECT 语句或者类型表的分层信息来指定要导出的数据。

权限

为下列其中一项:

- sysadm

- dbadm

或者对参与的每个表或视图的 CONTROL 或 SELECT 特权。对此功能施加基于标号的访问控制（LBAC）。如果导出的数据受到 LBAC 的保护，那么该数据可能会受到调用者的 LBAC 凭证的限制。

必需的连接

数据库。如果启用了隐式连接，那么将建立与缺省数据库的连接。

API 包含文件

```
db2ApiDf.h
```

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
    db2Export (
      db2Uint32 versionNumber,
      void * pParmStruct,
      struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ExportStruct
{
      char *piDataFileName;
      struct sqlu_media_list *piLobPathList;
      struct sqlu_media_list *piLobFileList;
      struct sqldcol *piDataDescriptor;
      struct sqllob *piActionString;
      char *piFileType;
      struct sqlchar *piFileTypeMod;
      char *piMsgFileName;
      db2int16 iCallerAction;
      struct db2ExportOut *poExportInfoOut;
      struct db2ExportIn *piExportInfoIn;
      struct sqlu_media_list *piXmlPathList;
      struct sqlu_media_list *piXmlFileList;
} db2ExportStruct;

typedef SQL_STRUCTURE db2ExportIn
{
      db2Uint16 *piXmlSaveSchema;
} db2ExportIn;

typedef SQL_STRUCTURE db2ExportOut
{
      db2Uint64 oRowsExported;
} db2ExportOut;

SQL_API_RC SQL_API_FN
    db2gExport (
      db2Uint32 versionNumber,
      void * pParmStruct,
      struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gExportStruct
{
      char *piDataFileName;
      struct sqlu_media_list *piLobPathList;
      struct sqlu_media_list *piLobFileList;
      struct sqldcol *piDataDescriptor;
      struct sqllob *piActionString;
      char *piFileType;
      struct sqlchar *piFileTypeMod;
      char *piMsgFileName;
```

```
      db2int16 iCallerAction;
      struct db2ExportOut *poExportInfoOut;
      db2Uint16 iDataFileNameLen;
      db2Uint16 iFileTypeLen;
      db2Uint16 iMsgFileNameLen;
      struct db2ExportIn *piExportInfoIn;
      struct sqlu_media_list *piXmlPathList;
      struct sqlu_media_list *piXmlFileList;
} db2gExportStruct;
```

db2Export API 参数

versionNumber

输入。指定作为第二个参数 pParmStruct 传递的结构的版本和发行版级别。

pParmStruct

输入。指向 db2ExportStruct 结构的指针。

pSqlca

输出。指向 sqlca 结构的指针。

db2ExportStruct 数据结构参数

piDataFileName

输入。一个字符串，它包含要将数据导出到的外部文件的路径和名称。

piLobPathList

输入。指向 sqlu_media_list 结构的指针，该结构的 media_type 字段设置为 SQLU_LOCAL_MEDIA，并且它的 sqlu_media_entry 结构列示客户机上用于存储 LOB 文件的路径。导出的 LOB 数据将平均分布在 sqlu_media_entry 结构中列示的所有路径中。

piLobFileList

输入。指向 sqlu_media_list 结构的指针，它的 media_type 字段设置为 SQLU_CLIENT_LOCATION，并且它的 sqlu_location_entry 结构包含基本文件名。

当耗尽了使用此列表中的第一个名称的名称空间时，API 将使用第二个名称，以此类推。在导出操作期间创建 LOB 文件时，按以下方法来构造文件名：将此列表中的当前基本名称追加至（piLobPathList 中的）当前路径，然后再追加三位数的序号和 .lob 扩展名。例如，如果当前 LOB 路径是 /u/foo/lob/path 目录、当前 LOB 文件名是 bar，并且设置了 LOBSINSEPFILES 文件类型修饰符，那么创建的 LOB 文件将为 /u/foo/LOB/path/bar.001.lob 和 /u/foo/LOB/path/bar.002.lob 等等。如果未设置 LOBSINSEPFILES 文件类型修饰符，那么将并置所有 LOB 文档，并放入同一个文件 /u/foo/lob/path/bar.001.lob 中。

piDataDescriptor

输入。指向 sqldcol 结构的指针，该结构用于指定输出文件的列名。dcolmeth 字段的值确定 EXPORT 实用程序如何解释此参数中提供的其余信息。此参数的有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是：

SQL_METH_N

名称。指定要在输出文件中使用的列名。

SQL_METH_D

缺省值。将在输出文件中使用表中的现有列名。在这种情况下，列数和列规范数组都将被忽略。列名是从 piActionString 中指定的 SELECT 语句的输出派生而来。

piActionString

输入。指向包含有效动态 SQL SELECT 语句的 sqllob 结构。该结构包含 4 字节长的字段，后面紧接组成 SELECT 语句的字符。SELECT 语句指定要从数据库中抽取并写入外部文件的数据。

（piDataDescriptor 中的）外部文件中的列与 SELECT 语句中的数据库列按照它们各自的列表/结构位置来进行匹配。从数据库中选择的第一列数据被放入外部文件的第一列，而它的列名是从外部列数组的第一个元素中获得的。

piFileType

输入。一个字符串，用来指示外部文件中数据的格式。受支持的外部文件格式是（这些文件格式是在 sqlutil 头文件中定义的）：

SQL_DEL

定界 ASCII，用于与 dBase、BASIC 和 IBM® Personal Decision Series 程序以及许多其他数据库管理器和文件管理器进行交换。

SQL_WSF

用于与 Lotus Symphony 和 1-2-3 程序进行交换的工作表格式。

SQL_IXF

PC 版本的集成交换格式，这是用于导出表中的数据的首选方法。然后，可以将用此文件格式导出的数据导入或装入同一个表中或者装入另一个数据库管理器表中。

piFileTypeMod

输入。指向 sqldcol 结构的指针，该结构包含一个两字节长的字段，后跟用来指定一个或多个处理选项的一组字符。如果此指针为 NULL，或者指向的结构未包含任何字符，那么会将此操作解释为选择缺省规范。

并不是所有选项都可用于所有受支持的文件类型。请参阅下面的相关链接: “EXPORT 实用程序的文件类型修饰符”。

piMsgFileName

输入。一个字符串，它包含由实用程序返回的错误消息、警告消息和参考消息的目标。它可以是操作系统文件或标准设备的路径和名称。如果该文件已存在，那么会追加信息。如果它不存在，那么会创建文件。

iCallerAction

输入。调用者请求的操作。有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是:

SQLU_INITIAL

初始调用。必须在首次调用 API 时使用此值。如果初始调用或任何后续调用已返回，但是要求进行调用的应用程序执行某些操作后才能完成所请求的导出操作，那么必须将调用者操作设置为下列其中一项:

SQLU_CONTINUE

继续处理。在初始调用返回的结果是实用程序要求用户输入（例如，

要求对磁带条件结束作出响应）后，只能在对 API 的后续调用中使用此值。它指定实用程序所请求的用户操作已完成，该实用程序可以继续处理初始请求。

SQLU_TERMINATE

终止处理。在初始调用返回的结果是实用程序要求用户输入（例如，要求对磁带条件结束作出响应）后，只能在对 API 的后续调用中使用此值。它指定未执行实用程序所请求的用户操作，并且实用程序将终止处理初始请求。

poExportInfoOut

指向 db2ExportOut 结构的指针。

piExportInfoIn

输入。指向 db2ExportIn 结构的指针。

piXmlPathList

输入。指向 sqlu_media_list 结构的指针，该结构的 media_type 字段设置为 SQLU_LOCAL_MEDIA，并且它的 sqlu_media_entry 结构列示客户机中用于存储 XML 文件的路径。导出的 XML 数据将平均分布在 sqlu_media_entry 结构中列示的所有路径中。

piXmlFileList

输入。指向 sqlu_media_list 结构的指针，它的 media_type 字段设置为 SQLU_CLIENT_LOCATION，并且它的 sqlu_location_entry 结构包含基本文件名。

当耗尽了使用此列表中的第一个名称的名称空间时，API 将使用第二个名称，以此类推。在导出操作期间创建 XML 文件时，按以下方法来构造文件名：将此列表中的当前基本名称追加至（piXmlFileList 中的）当前路径后，然后再追加三位数的序号和 .xml 扩展名。例如，如果当前 XML 路径是 /u/foo/xml/path 目录，当前 XML 文件名是 bar，并且设置了 XMLINSEPFILES 文件类型修饰符，那么创建的 XML 文件将为 /u/foo/xml/path/bar.001.xml 和 /u/foo/xml/path/bar.002.xml 等等。如果未设置 XMLINSEPFILES 文件类型修饰符，那么将并置所有 XML 文档，并放入同一个文件 /u/foo/xml/path/bar.001.xml 中。

db2ExportIn 数据结构参数

piXmlSaveSchema

输入。指示应将 XML 模式的 SQL 标识保存在导出的数据文件中，该模式用来验证导出的每个 XML 文档。它的值可能是 TRUE 或 FALSE。

db2ExportOut 数据结构参数

oRowsExported

输出。返回已导出到目标文件的记录数。

特定于 db2gExportStruct 数据结构的参数

iDataFileNameLen

输入。一个两字节的无符号整数，表示数据文件名的长度（以字节计）。

iFileTypeLen

输入。一个两字节的无符号整数，表示文件类型的长度（以字节计）。

iMsgFileNameLen

输入。一个两字节的无符号整数，表示消息文件名的长度（以字节计）。

使用说明

在开始执行导出操作前，必须采用下面任一方法来完成所有表操作并释放所有锁定:

- 关闭所有已打开的使用 WITH HOLD 子句定义的游标，并通过执行 COMMIT 语句来落实数据更改。
- 通过执行 ROLLBACK 语句来回滚数据更改。

可以在 SELECT 语句中使用表别名。

放入消息文件中的消息包括从消息检索服务中返回的信息。每条消息都另起一行。

如果 EXPORT 实用程序产生警告，那么会将消息写入消息文件，如果未指定消息文件，那么会写入标准输出。

如果外部列名数组 piDataDescriptor 中的列数（sqldcol 结构的 dcolnum 字段）与 SELECT 语句生成的列数不相等，那么会发出警告消息。在这种情况下，写入外部文件的列数是这两个数中的较小者。多出的数据库列或外部列名未用来生成输出文件。

如果手动绑定了 db2uexpm.bnd 模块或已交付的任何其他 .bnd 文件，那么不能对绑定程序使用格式选项。

可以使用 DB2 Connect 从 DRDA 服务器（例如，DB2 z/OS® 和 OS/390 版、DB2 VM 和 VSE 版以及 DB2 System i™ 版）中导出表。仅支持 PC/IXF 导出。

应该使用 PC/IXF 导入来在数据库之间移动数据。如果包含行分隔符的字符数据已导出到定界 ASCII（DEL）文件并且已由文本传输程序处理，那么包含行分隔符的字段将收缩或展开。

从 AIX 系统中调用 EXPORT 实用程序时，该程序将不会创建由多部分组成的 PC/IXF 文件。

当将单个数据库表的内容导出到 PC/IXF 文件，并且 piActionString 参数以 SELECT * FROM tablename 开头，piDataDescriptor 参数指定了缺省名称时，表的索引定义包括在 PC/IXF 文件中。如果 piActionString 的 SELECT 子句包括连接，那么不会保存视图的索引。piActionString 参数中的 WHERE 子句、GROUP BY 子句或 HAVING 子句将不会阻止保存索引。在所有这些情况下，从类型表中导出时，必须导出整个层次结构。

如果所提供的 SELECT 语句采用以下格式: SELECT * FROM tablename，那么 EXPORT 实用程序会将表的 NOT NULL WITH DEFAULT 属性存储在一个 IXF 文件中。

当导出类型表时，只能通过指定目标表名和 WHERE 子句来表示子查询语句。而在导出层次结构时，不能指定全查询和 select-statement。

对于不是 IXF 的文件格式，建议您指定遍历顺序列表，这是因为它会告诉 DB2 如何遍历层次结构以及要导出哪些子表。如果未指定此列表，那么将导出层次结构中的所有表，并且缺省顺序为 OUTER 顺序。还可以使用缺省顺序，也就是由 OUTER 函数给定的顺序。

注: 在导入操作期间使用同样的遍历顺序。LOAD 实用程序不支持装入层次结构或子层次结构。

REXX™ API 语法

```
EXPORT :stmt TO datafile OF filetype
[MODIFIED BY :filetmod] [USING :dcoldata]
MESSAGES msgfile [ROWS EXPORTED :number]


CONTINUE EXPORT


STOP EXPORT
```

REXX API 参数

stmt 包含有效动态 SQL SELECT 语句的 REXX 主变量。该语句指定要从数据库中抽取的数据。

datafile
要将数据导出到的文件的名称。

filetype
导出文件中数据的格式。受支持的文件格式包括:

DEL 定界 ASCII

WSF 工作表格式

IXF PC 版本的集成交换格式。

filetmod
包含其他处理选项的主变量。

dcoldata
一个复合 REXX 主变量，它包含要在导出文件中使用的列名。在下列各项中，XXX 表示主变量的名称:

XXX.0 列数（变量的其余部分中包含的元素数）。

XXX.1 第一个列名。

XXX.2 第二个列名。

XXX.3 以此类推。

如果此参数为 NULL，或者尚未指定 dcoldata 的值，那么该实用程序将使用数据库表中的列名。

msgfile
要将错误消息和警告消息发送至的文件、路径或设备的名称。

number
一个主变量，它将包含导出的行数。

导出会话 - CLP 示例

示例 1
以下示例说明如何以 IXF 输出格式将 SAMPLE 数据库（用户必须连接至的数据库）的

STAFF 表中的信息导出至 myfile.ixf。如果未通过 DB2 Connect 进行数据库连接，那么索引定义（如果存在）将存储在输出文件中；否则仅存储数据:

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

示例 2

以下示例说明如何以 IXF 输出格式将 SAMPLE 数据库（用户必须连接至的数据库）的 STAFF 表中有关 Department 20 的职员的信息导出至 awards.ixf。

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
   where dept = 20
```

示例 3以下示例说明如何将 LOB 导出到 DEL 文件:

```
db2 export to myfile.del of del lobs to mylobs/
      lobfile lobs1, lobs2 modified by lobsinfile
select * from emp_photo
```

示例 4

以下示例说明如何将 LOB 导出到 DEL 文件，对可能无法装入到第一个目录中的文件指定第二个目录:

```
db2 export to myfile.del of del
         lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
   select * from emp_photo
```

示例 5以下示例说明如何将数据导出到 DEL 文件，将单引号用作字符串定界符，分号用作列定界符，以及逗号用作小数点。在将数据导入回数据库时应使用同一约束:

```
db2 export to myfile.del of del
         modified by chardel'' coldel; decpt,
         select * from staff
```

第 3 章 import 实用程序

导入概述

import 实用程序会使用 SQL INSERT 语句向表、类型表或视图填充数据。如果接收已导入数据的表或视图已包含数据，那么输入数据可替换现有数据，也可追加至现有数据。

与 export 一样，import 是相对简单的数据移动实用程序。可通过下列方法激活它：通过控制中心、发出 CLP 命令、调用 ADMIN_CMD 存储过程、调用其 API（db2Import）或通过用户应用程序。

import 支持多种数据格式，并且可与多种功能配合使用:

- import 支持 IXF、WSF、ASC 和 DEL 数据格式。
- import 可与文件类型修饰符配合使用以定制导入操作。
- import 可用于移动分层数据和类型表。
- import 可记录所有活动、更新索引、验证约束和激发触发器。
- import 允许您在要向其插入数据的表或视图中指定列名。
- import 可与 DB2 Connect 配合使用。

导入方式

导入可使用五种方式，它们用于确定导入数据的方法。前三种方式为 INSERT、INSERT_UPDATE 和 REPLACE，在目标表已存在的情况下使用。这三种方式都支持 IXF、WSF、ASC 和 DEL 数据格式。但是，只有 INSERT 和 INSERT_UPDATE 可与昵称配合使用。

表 11.

方式	最佳实践用法
INSERT	将输入数据插入到目标表中而不更改现有数据
INSERT_UPDATE	使用输入行的值更新具有匹配主键值的行 如果没有匹配行，那么会将已导入行插入到表中
REPLACE	删除所有现有数据并插入已导入数据，同时保留表和索引定义

另外两种方式为 REPLACE_CREATE 和 CREATE，在目标表不存在时使用。它们只能与 PC/IXF 格式的输入文件配合使用，此格式包含要创建的表的结构化描述。如果对象表具有自身以外的任何从属，那么不能以这些方式执行导入。

注: 不建议使用导入的 CREATE 和 REPLACE_CREATE 方式。请改用 db2look 实用程序。

表 12.

方式	最佳实践用法
REPLACE_CREATE	删除所有现有数据并插入已导入数据，同时保留表和索引定义 如果目标表和索引不存在，那么创建目标表和索引
CREATE	创建目标表和索引 可指定在其中创建新表的表空间名称

导入的工作方式

导入所需的步骤数和时间量取决于要移动的数据量和指定的选项。导入操作遵循下列步骤:

1. 锁定表
   根据您是否允许对表进行并行访问，导入会获取对现有目标表的独占（X）或非独占（IX）锁定。
2. 查找和检索数据
   导入使用 FROM 子句来查找输入数据。如果命令指示 XML 或 LOB 数据存在，那么导入会查找此数据。
3. 插入数据
   导入会替换现有数据或将新的数据行添加至表。
4. 检查约束和激发触发器
   写入数据后，导入会确保每个已插入行符合针对目标表定义的约束。有关被拒绝行的信息将写至消息文件。导入还会激发现有触发器。
5. 落实操作
   导入会保存所作更改并释放针对目标表的锁定。还可指定在导入期间定期落实。

下列各项是基本导入操作所必需的:

- 输入文件的路径和名称
- 目标表或视图的名称或别名
- 输入文件中的数据格式
- 用于导入数据的方法
- 导入分层数据时采用的遍历顺序
- 导入类型表时采用的子表列表

其他选项

许多选项允许您定制导入操作。可在 MODIFIED BY 子句中指定文件类型修饰符以更改数据格式，告诉 import 实用程序如何处理数据及改进性能。

缺省情况下，import 实用程序直到导入成功结束才执行落实，但某些 ALLOW WRITE ACCESS 导入例外。这会改变导入速度，但考虑到并行性、可重新启动性及活动日志空间注意事项，最好指定落实在导入期间进行。其中一种方法是将 **COMMITCOUNT** 参数设置为“automatic”，这会指示导入在内部确定何时应执行落实。或者，您可将 **COMMITCOUNT** 设置为特定数字，以指示导入在达到指定的已导入记录数时执行一次落实。

有几种方法可用来改进导入性能。因为 import 实用程序是嵌入式 SQL 应用程序并且以内部方式执行 SQL 访存，所以应用于 SQL 操作的优化会同时应用于导入。可使用 compound 文件类型修饰符以便一次插入指定数目的行，而不是按缺省方式逐行插入。如果预计导入期间会生成大量警告（并因此导致操作变慢），还可指定 norowwarnings 文件类型修饰符以抑制有关被拒绝行的警告。

消息文件

在导入期间，将编写标准 ASCII 文本消息文件以包含与该操作相关联的错误消息、警告消息和参考消息。如果该实用程序是通过应用程序编程接口（API）db2Import 调用的，那么必须使用 **MESSAGES** 参数预先指定这些文件的名称，否则此参数是可选参数。使用消息文件可以很方便地监视导入进度，原因是您可在导入正在进行时访问消息文件。如果导入操作失败，那么可使用消息文件来确定重新启动位置，原因是消息文件会指示成功导入的最后一行。

注: 如果针对远程数据库的导入操作生成的输出消息量超过 60 KB，那么该实用程序会保留前 30 KB 和后 30 KB。

使用 IMPORT 所需的特权和权限

特权使用户能够创建或访问数据库资源。权限级别提供了对特权、较高级别数据库管理器维护和实用程序操作进行分组的方法。这两者一起用于控制对数据库管理器及其数据库对象的访问。用户只能访问那些他们具有相应授权（即必需的特权或权限）的对象。

有了 SYSADM 和 DBADM 权限，您可以执行任何类型的导入操作。下表列示对应每个参与表、视图或昵称以允许您执行相应导入类型的其他权限。

表 13.

方式	必需权限
INSERT	CONTROL 或 INSERT 和 SELECT
INSERT_UPDATE	CONTROL 或 INSERT、SELECT、UPDATE、DELETE
REPLACE	CONTROL 或 INSERT、SELECT、DELETE
REPLACE_CREATE	目标表存在时：CONTROL 或 INSERT、SELECT、DELETE 目标表不存在时：CREATETAB（针对数据库）、USE（针对表空间），以及 模式不存在时：IMPLICIT_SCHEMA（针对数据库）或 模式存在时：CREATEIN（针对模式）
CREATE	CREATETAB（针对数据库）和 USE（针对表空间），以及 模式不存在时：IMPLICIT_SCHEMA（针对数据库）或 模式存在时：CREATEIN（针对模式）

注: 不推荐使用 IMPORT 命令的 CREATE 和 REPLACE_CREATE 选项，将来的发行版中可能会除去这两个选项。

同样，要对表使用 REPLACE 或 REPLACE_CREATE 选项，会话授权标识必须有权废弃该表。

如果要导入至层次结构，那么必需权限也取决于导入方式。对于现有层次结构，针对层次结构中每个子表的 CONTROL 特权足以执行 REPLACE 操作。对于不存在的层次结构，针对层次结构中每个子表的 CONTROL 特权加上 CREATETAB 和 USE 足以执行 REPLACE_CREATE 操作。

此外，导入到定义了基于标号的访问控制（LBAC）安全标号的表中时，有一些注意事项。要将数据导入到包含受保护列的表中，会话授权标识必须拥有允许对该表中所有受保护列执行写访问的 LBAC 凭证。要将数据导入到包含受保护行的表中，必须已将保护该表的安全策略所包含的写访问安全标号授予会话授权标识。

导入数据

import 实用程序将具有受支持文件格式的外部文件中的数据插入到表、层次结构、视图或昵称中。另一种速度更快的方法是使用 load 实用程序，但 load 实用程序不支持在层次结构级别装入数据。

在调用 import 实用程序前，必须连接至（或者能够隐式连接至）要导入数据的数据库。如果启用了隐式连接，那么将建立与缺省数据库的连接。实用程序必须借助通过引擎（而不是通过 DB2 Connect 网关或回送环境）的直接连接从 DB2 Linux 版、UNIX 版或 Windows 版客户机访问 DB2 Linux 版、UNIX 版或 Windows 版数据库服务器。由于实用程序将发出 COMMIT 或 ROLLBACK 语句，所以在调用导入之前，应该通过发出 COMMIT 或 ROLLBACK 操作以完成所有事务并释放所有锁定。

注: 不推荐使用 IMPORT 命令的 CREATE 和 REPLACE_CREATE 选项，将来的发行版中可能会除去这两个选项。

import 实用程序存在下列限制:

- 如果现有表是一个父表，并且它包含的主键被从属表中的外键引用，那么不能替换此表的数据，而只能追加数据。
- 不能执行导入替换操作来将数据导入到以立即刷新方式定义的具体化查询表的基础表中。
- 不能将数据导入到系统表、总结表或其他带有结构化类型列的表中。
- 不能将数据导入到已声明临时表中。
- 不能通过 import 实用程序创建视图。
- 根据 PC/IXF 文件创建表时，不会保留引用约束和外键定义。如果数据先前是使用 SELECT * 导出的，那么会保留主键定义。
- 由于 import 实用程序会生成自己的 SQL 语句，所以在某些情况下可能会超过最大语句大小（即 2 MB）。
- 不能使用 CREATE 或 REPLACE_CREATE 导入选项重新创建分区表或多维集群表（MDC）。
- 不能重新创建包含 XML 列的表。
- 不能导入已加密的数据。
- 导入替换操作不能识别 Not Logged Initially 子句。IMPORT 命令的 REPLACE 选项不能识别 CREATE TABLE 语句的 NOT LOGGED INITIALLY (NLI) 子句或 ALTER TABLE 语句的 ACTIVATE NOT LOGGED INITIALLY 子句。如果包含

REPLACE 操作的导入操作与调用了 NLI 子句的 CREATE TABLE 或 ALTER TABLE 语句在同一事务内执行，那么此导入操作不能识别该 NLI 子句。将记录所有插入操作。

变通方法 1: 使用 DELETE 语句删除表的内容，然后使用 INSERT 语句调用导入操作。

变通方法 2: 废弃然后重新创建该表，接着使用 INSERT 语句调用导入操作。

以下局限性适用于 import 实用程序: 如果导入操作针对远程数据库生成的输出消息量超过 60 KB 则 import 实用程序会保留前 30 KB 和后 30 KB。

可以通过命令行处理器（CLP）、“控制中心”中的“导入表”笔记本或者从客户机应用程序中调用 db2Import 应用程序编程接口（API）来调用 import 实用程序。

使用“导入表”笔记本

1. 在“控制中心”中，展开对象树，直到找到“表”文件夹为止。
2. 单击“表”文件夹。所有现有表都会显示在窗口右边的窗格（内容窗格）中。
3. 在内容窗格中右键单击想要的表，然后从弹出菜单中选择“导入”。这就打开了“导入表”笔记本。

“控制中心”的联机帮助工具提供了有关“导入表”笔记本的详细信息。

通过使用 CLP 发出 IMPORT 命令

这是一个非常简单的导入操作，只需要您指定输入文件、文件格式、导入方式和目标表（或要创建的表名）。

例如，要从 CLP 导入数据，请输入 IMPORT 命令:

```
db2 import from filename of fileformat import_mode into table
```

其中 filename 是包含要导入的数据的输入文件的名称，ixf 是文件格式，insert 是方式，而 table 是要向其插入数据的表名。

但是，您可能还想指定用于写入警告消息和错误消息的消息文件。为此，添加 **MESSAGES** 参数和消息文件名称，所以命令为如下所示:

```
db2 import from filename of fileformat messages messagefile import_mode into table
```

有关完整语法和用法信息，请参阅“IMPORT 命令”。

导入 XML 数据

通过对 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据对象使用表名或昵称，可使用 IMPORT 实用程序将 XML 数据导入到 XML 表列中。

将数据导入到 XML 表列中时，可以使用 XML FROM 选项来指定一个或多个输入 XML 数据文件的路径。例如，对于先前已导出的 XML 文件“/home/user/xmlpath/xmldocs.001.xml”，可以使用下列命令将数据导入回表中。

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

针对模式验证插入的文档

XMLVALIDATE 选项允许在导入 XML 文档时针对 XML 模式验证这些文档。在以下示例中，将针对导出 XML 文档时保存的模式信息验证入局 XML 文档:

```
IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
    USING XDS INSERT INTO USER.T1
```

指定解析选项

可以使用 XMLPARSE 选项来指定是保留还是去掉已导入的 XML 文档中的空格。在以下示例中，将针对导出 XML 文档时保存的 XML 模式信息验证已导入的所有 XML 文档，并在保留空格的情况下解析这些文档。

```
   IMPORT FROM t1export.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
WHITESPACE XMLVALIDATE USING XDS INSERT INTO USER.T1
```

已导入表重新创建

可以使用 import 实用程序的 CREATE 方式来重新创建先前通过 export 实用程序保存的表。但是，处理时有很多局限性，原因是输入表的许多属性未保留下来。

为了让导入能够重新创建该表，导出操作必须符合某些要求。原始表必须已导出至 IXF 文件。如果使用 DEL 或 ASC 文件格式导出文件，那么输出文件不会包含目标表的描述，但它们包含记录数据。要使用以这些文件格式存储的数据来重新创建表，请创建目标表，然后使用 load 实用程序或 import 实用程序根据这些文件填充该表。可使用 db2look 实用程序来捕获原始表定义，并生成相应的数据定义语言（DDL）。同时，导出期间使用的 SELECT 语句只能包含特定操作字符串。例如， SELECT 子句中不能使用任何列表，只允许使用 SELECT *。

注: 不建议使用导入的 CREATE 方式。请使用 db2look 实用程序来捕获并重新创建表。

保留的属性

重新创建的表将保留原始表的下列属性:

- 主键名和定义
- 列信息，包括:
  - 列名
  - 列数据类型，包括用户定义的单值类型（它们将作为基本类型保留）
  - 标识属性
  - 长度（lob_file 类型除外）
  - 代码页（如果适用）
  - 标识选项
  - 列是定义为可空还是不可空
  - 常量的缺省值（如果存在），但不包括其他类型的缺省值
- 索引信息，包括:
  - 索引名
  - 索引创建者名
  - 列名以及每列是按升序还是降序排序
  - 索引是否被定义为唯一索引

- 索引是否是集群索引
- 索引是否允许逆向扫描
- PCTFREE 值
- MINPCTUSED 值

注: 如果索引中的列名包含字符 - 或 +，那么不会保留任何索引信息，并且返回 SQL27984W。

丢失的属性

重新创建的表不会保留原始表的某些属性，包括:

- 源是常规表、具体化查询表（MQT）、视图还是任何或所有这些源中的一部分列
- 唯一约束及其他类型的约束或触发器（不包括主键约束）
- 表信息，包括:
  - MQT 定义（如果适用）
  - MQT 选项（如果适用）
  - 表空间选项; 但是，可以通过 IMPORT 命令指定此信息
  - 多维集群（MDC）维
  - 分区表维
  - 表分区键
  - NOT LOGGED INITIALLY 属性
  - 检查约束
  - 表代码页
  - 受保护的表属性
  - 表或值压缩选项
- 列信息，包括:
  - 除常量值以外的任何缺省值
  - LOB 选项（如果存在）
  - XML 属性
  - CREATE TABLE 语句的 REFERENCE 子句（如果存在）
  - 引用约束（如果存在）
  - 检查约束（如果存在）
  - 生成列选项（如果存在）
  - 依赖于数据库作用域序列的列
- 索引信息，包括:
  - INCLUDE 列（如果存在）
  - 索引名（如果该索引是主键索引）
  - 如果该索引是主键索引，那么按照键的降序排序（缺省顺序是升序）
  - 包含十六进制值 0x2B 或 0x2D 的索引列名
  - 在进行代码页转换后包含超过 128 个字节的索引名
  - PCTFREE2 值
  - 唯一约束

注: 此列表并未涵盖所有情况，请谨慎使用。

如果导入失败并返回 SQL3311N，您仍然可以使用文件类型修饰符 `forcecreate` 来重新创建该表。此修饰符允许您创建信息缺少或信息有限的表。

类型表导入注意事项

可以使用 import 实用程序将数据移入移出类型表，同时保留数据预先存在的层次结构。必要时还可使用导入来创建表层次结构和类型层次结构。

通过使用特定遍历顺序并在导出操作期间创建中间平面文件，可将数据从类型表的一个层次结构移至另一个层次结构。而 import 实用程序会控制要通过 CREATE、INTO *table-name*、UNDER 和 AS ROOT TABLE 参数移动的层次结构的大小和放置。同时，导入会控制放置在目标数据库中的内容。例如，可在每个子表名称的结尾指定属性列表，以限制移至目标数据库的属性。如果未使用属性列表，那么将移动每个子表中的所有列。

表重新创建

能够执行的导入类型取决于输入文件的文件格式。处理 ASC、DEL 或 WSF 数据时，目标表或层次结构必须存在，才能导入数据。但是，如果指定了导入 CREATE 操作，即使表或层次结构尚未存在，也可导入 PC/IXF 文件中的数据。必须注意的是，如果指定了 CREATE 选项，那么导入不能改变子表定义。

遍历顺序

输入文件中包含的遍历顺序允许保留数据中的层次结构。因此，在调用 export 实用程序和 import 实用程序时，必须使用相同的遍历顺序。

对于 PC/IXF 文件格式，用户只需要指定目标子表名，并使用文件中存储的缺省遍历顺序。

当对类型表使用 CREATE 以外的其他选项时，遍历顺序列表允许用户指定遍历顺序。用户指定的遍历顺序必须与导出操作期间使用的遍历顺序相匹配。只要符合下列条件，import 实用程序会保证将数据准确移动至目标数据库:

- 源数据库和目标数据库上的子表定义完全相同
- 源数据库和目标数据库上的子表间的层次结构关系完全相同
- 遍历顺序完全相同

尽管在定义遍历顺序时确定了遍历层次结构的起始点和路径，但必须先遍历至每个分支的底部才能在层次结构中启动下一个分支。import 实用程序在指定的遍历顺序中查找是否存在违反此条件的情况。

示例

本节中的示例基于以下层次结构及四种有效遍历顺序:

- Person -> Employee -> Manager -> Architect -> Student
- Person -> Student -> Employee -> Manager -> Architect
- Person -> Employee -> Architect -> Manager -> Student
- Person -> Student -> Employee -> Architect -> Manager

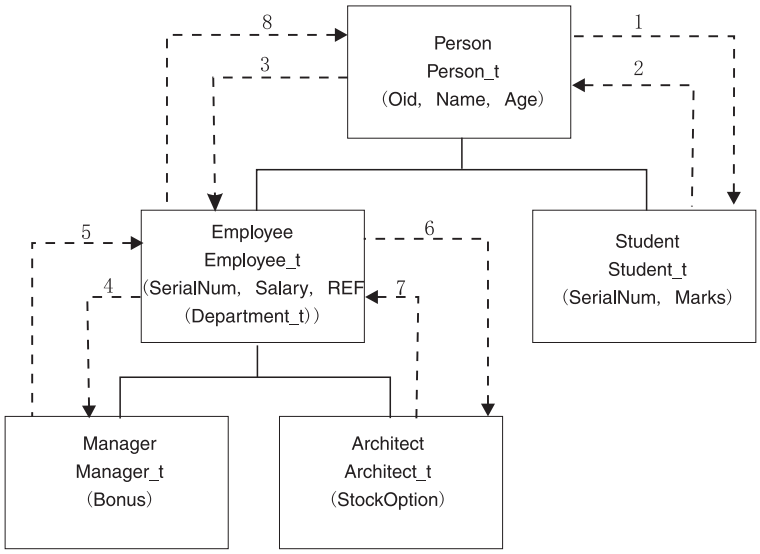


图 2. 层次结构示例

示例 1

要使用导入重新创建整个层次结构（包括在由先前的导出操作创建的数据文件 entire_hierarchy.ixf 中），应输入下列命令:

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.ixf OF IXF CREATE INTO
        HIERARCHY STARTING Person AS ROOT TABLE
```

将在层次结构中创建不存在的每个类型。如果这些类型已存在，那么它们在目标数据库和源数据库中的定义必须相同。如果不相同，将返回 SQL 错误（SQL20013N）。因为要创建新层次结构，所以在数据文件中定义的要移至目标数据库（Target_db）的子表都不能存在。将在源数据库层次结构中创建每个表。源数据库中的数据将导入到目标数据库的正确子表中。

示例 2

要重新创建源数据库的整个层次结构并将其导入到目标数据库中，同时仅保留所选数据，应输入以下命令:

```
DB2 CONNECT TO Target_db
DB2 IMPORT FROM entire_hierarchy.del OF DEL INSERT INTO (Person,
        Employee(Salary), Architect) IN HIERARCHY (Person, Employee,
        Manager, Architect, Student)
```

目标表 PERSON、EMPLOYEE 和 ARCHITECT 必须都存在。数据将导入到 PERSON、EMPLOYEE 和 ARCHITECT 子表中。即，会将:

- PERSON 中的所有列导入到 PERSON 中
- PERSON 中的所有列及 EMPLOYEE 中的 SALARY 导入到 EMPLOYEE 中
- PERSON 中的所有列、EMPLOYEE 中的 SALARY 及 ARCHITECT 中的所有列导入到 ARCHITECT 中

列 SerialNum 和 REF(Employee_t) 不会导入到 EMPLOYEE 或其子表（即 ARCHITECT，它是数据将导入其中的唯一子表）中。

注: 因为 ARCHITECT 是 EMPLOYEE 的子表，并且对 EMPLOYEE 指定的唯一导入列为 SALARY，所以 SALARY 同时也是导入到 ARCHITECT 中的唯一特定于

EMPLOYEE 的列。即，SerialNum 和 REF(Employee_t) 列不会导入到 EMPLOYEE 或 ARCHITECT 行中。
MANAGER 和 STUDENT 表的数据不会导入。

示例 3

此示例显示如何从常规表导出以及在层次结构中作为单个子表导入。EXPORT 命令作用于常规（非类型）表，所以数据文件中没有 Type_id 列。文件类型修饰符 no_type_id 用于指示这种情况，所以 import 实用程序不应将第一列作为 Type_id 列。

```
DB2 CONNECT TO Source_db
DB2 EXPORT TO Student_sub_table.del OF DEL SELECT * FROM
   Regular_Student
DB2 CONNECT TO Target_db
DB2 IMPORT FROM Student_sub_table.del OF DEL METHOD P(1,2,3,5,4)
   MODIFIED BY NO_TYPE_ID INSERT INTO HIERARCHY (Student)
```

在此示例中，目标表 STUDENT 必须存在。因为 STUDENT 是子表，所以修饰符 no_type_id 用于指示第一列中没有 Type_id。但是，除了 STUDENT 表中存在的所有其他属性之外，还必须确保存在 Object_id 列。Object-id 应该是导入至 STUDENT 表的每一行中的第一列。METHOD 子句使最后两个属性的顺序反向。

LBAC 保护的数据导入注意事项

要成功导入到带有受保护行的表中，必须具有 LBAC（基于标号的访问控制）凭证。对于当前与目标表相关联的安全策略，必须同时提供有效安全标号或者可转换为有效标号的安全标号。

如果没有有效 LBAC 凭证，导入会失败并返回错误（SQLSTATE 42512）。如果输入数据未包含安全标号或该安全标号未使用其内部二进制格式，那么可使用若干文件类型修饰符以允许继续导入。

将数据导入到带有受保护行的表中时，目标表必须具有数据类型为 DB2SECURITYLABEL 的一列。如果输入数据行未包含该列的值，那么该行会被拒绝，除非在导入命令中指定了 usedefaults 文件类型修饰符，此时将使用您拥有的安全策略（用于保护表）中对应写访问权的安全标号。如果没有对应写访问权的安全标号，那么该行将被拒绝，并且处理会继续进至下一行。

如果要将数据导入到带有受保护行的表中，并且输入数据包括数据类型为 DB2SECURITYLABEL 列的值，那么将数据插入到该表中时遵循相同规则。如果用于保护要导入行的安全标号（数据文件的该行中的安全标号）就是您能够写入的安全标号，那么该安全标号将用于保护该行。（换言之，它将写入数据类型为 DB2SECURITYLABEL 的列。）如果您无法写入受该安全标号保护的行，那么产生的结果取决于保护源表的安全策略的创建方式:

- 如果创建该策略的 CREATE SECURITY POLICY 语句包含 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么插入操作将会失败并返回错误。
- 如果 CREATE SECURITY POLICY 语句未包含该选项或者包含 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么数据文件中对应该行的安全标号将被忽略，而您拥有的对应写访问权的安全标号将用于保护该行。在这种情况下，不会发出任何错误或警告。如果没有对应写访问权的安全标号，那么该行将被拒绝，并且处理会继续进至下一行。

定界符注意事项

将数据导入到数据类型为 DB2SECURITYLABEL 的列中时，缺省情况下会将数据文件中的值视为组成该安全标号的内部表示的实际字节数。但是，某些原始数据可能包含换行符，它们可能会被 IMPORT 命令错误地解释为行定界符。如果发生此问题，请使用 delprioritychar 文件类型修饰符以确保字符定界符优先于行定界符。使用 delprioritychar 时，包含在字符定界符中的任何记录或列定界符不会被识别为定界符。即使没有任何值包含换行符，使用 delprioritychar 文件类型修饰符也很安全，但这会稍微降低导入的速度。

如果要导入的数据为 ASC 格式，那么您可能想要执行一个额外步骤，以避免已导入安全标号和安全标号名称中包含尾部空格。ASCII 格式将列位置用作定界符，所以导入到变长字段中时可能出现此情况。请使用 striptblanks 文件类型修饰符截断所有尾部空格。

非标准安全标号值

还可导入这样的数据文件，其安全标号值是包含安全标号中的组件的值的字符串，例如，*S:(ALPHA,BETA)*。为此，必须使用文件类型修饰符 seclabelchar。使用 seclabelchar 时，数据类型为 DB2SECURITYLABEL 的列的值将被视为字符串常量，包含对应安全标号的字符串格式的安全标号。如果字符串的格式不正确，那么不会插入该行，并且会返回警告（SQLSTATE 01H53）。如果该字符串未表示保护表的安全策略中的有效安全标号，那么不会插入该行，并且会返回警告（SQLSTATE 01H53）。

还可导入数据文件，其安全标号列的值是安全标号名称。要导入此类文件，必须使用文件类型修饰符 seclabelname。使用 seclabelname 时，数据类型为 DB2SECURITYLABEL 的所有列值将被视为包含现有安全标号名称的字符串常量。如果带有对应保护表的安全策略的指示名称的安全标号不存在，那么不会插入该行，并且会返回警告（SQLSTATE 01H53）。

示例

对于所有示例，输入数据文件 myfile.del 将使用 DEL 格式。数据会导入到使用以下语句创建的表 REPS 中：

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

对于此示例，假定输入文件包含缺省格式的安全标号：

```
db2 import from myfile.del of del modified by delprioritychar insert into reps
```

对于此示例，假定输入文件包含安全标号字符串格式的安全标号：

```
db2 import from myfile.del of del modified by seclabelchar insert into reps
```

对于此示例，假定输入文件包含安全标号列的安全标号名称：

```
db2 import from myfile.del of del modified by seclabelname insert into reps
```

缓冲插入导入

在分区数据库环境中，可以允许 import 实用程序使用缓冲插入功能。这将减少导入数据时进行的消息传递工作，从而提高性能。

仅当您不关心错误报告时才应启用缓冲插入选项，原因是启用此选项后不会返回有关失败的缓冲插入的详细信息。

使用缓冲插入后，导入会将缺省 **WARNINGCOUNT** 值设置为 1。因此，如果任何行被拒绝，操作将失败。如果拒绝某个记录，那么该实用程序还会回滚当前事务。可以使用已落实记录数来确定已成功插入数据库的记录。仅当指定了 COMMITCOUNT 选项时，已落实记录数才会是非零值。

如果在导入命令上显式指定了另一 **WARNINGCOUNT** 值，并且拒绝某些行，那么实用程序的行总结输出将是不正确的。这是由以下两方面原因引起的：缓冲插入功能使用异步错误报告功能；如果在插入一组行期间检测到错误，那么会导致回退该组中所有的行。由于此实用程序无法可靠地报告拒绝的输入记录，所以难以确定已落实的记录以及需要重新插入到数据库中的记录。

使用 DB2 Bind 实用程序来请求执行缓冲插入功能。必须使用 INSERT BUF 选项对数据库重新绑定导入包 db2uimpm.bnd。例如：

```
db2 connect to your_database
   db2 bind db2uimpm.bnd insert buf
```

缓冲插入功能无法与 INSERT_UPDATE 方式下的导入操作配合使用。绑定文件 db2uImpInsUpdate.bnd 将强制实施此限制。不应使用 INSERT BUF 选项绑定此文件。这会导致 INSERT_UPDATE 方式下的导入操作失败。绑定新文件不会影响 INSERT、REPLACE 或 REPLACE_CREATE 方式下的导入操作。

标识列导入注意事项

无论输入数据是否具有标识列值，都可以使用 import 实用程序将数据导入到包含标识列的表中。

如果未使用与标识相关的文件类型修饰符，那么该实用程序会遵循下列规则来工作：

- 如果标识列是 GENERATED ALWAYS 列，那么每当输入文件中的相应行缺少标识列值，或者显式指定了 NULL 值时，会为表行生成标识值。如果对标识列指定了非空值，那么会拒绝该行（SQL3550W）。
- 如果标识列是 GENERATED BY DEFAULT 列，那么 import 实用程序会使用用户提供的值（如果提供了这些值）；如果缺少数据或者显式指定了 NULL，那么会生成值。

除了通常对标识列数据类型（即 SMALLINT、INT、BIGINT 或 DECIMAL）的值执行的验证操作以外，import 实用程序不会对用户提供的标识值执行任何其他的验证操作。不报告重复值。此外，在将数据导入到带有标识列的表中时，不能使用 `compound=x` 修饰符。

有两种方法可用来简化将数据导入到包含标识列的表中的操作：`identitymissing` 和 `identityignore` 文件类型修饰符。

在没有标识列的情况下导入数据

如果输入数据文件未包含任何标识列值（甚至未包含 NULL 值），那么 `identitymissing` 修饰符可以使您更方便地导入带有标识列的表。例如，考虑使用以下 SQL 语句定义的表：

```
create table table1 (c1 char(30),
                     c2 int generated by default as
                        identity,
                     c3 real,
                     c4 char(1))
```

用户可能想要将数据从文件（import.del）导入到 TABLE1 中，并且此数据可能是从没有标识列的表中导出的。下面是此类文件的一个示例:

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

导入此文件的一种方法是通过 IMPORT 命令显式列示所要导入的列，如下所示:

```
db2 import from import.del of del replace into table1 (c1, c3, c4)
```

但是，对于包含许多列的表来说，此语法难以使用并且容易出错。导入该文件的另一种方法是使用 identitymissing 文件类型修饰符，如下所示:

```
db2 import from import.del of del modified by identitymissing
   replace into table1
```

在带有标识列的情况下导入数据

identityignore 修饰符在某些方面与 identitymissing 修饰符相反。它指示 import 实用程序: 即使输入数据文件包含标识列数据，也应该忽略该数据，并且应该为每一行生成标识值。例如，用户可能想将以下数据按照上述定义从文件（import.del）导入到 TABLE1 中:

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

如果用户提供的值 1、2 和 3 未用于标识列，那么该用户可以发出以下 IMPORT 命令:

```
db2 import from import.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
```

同样，如果该表包含许多列，那么此方法可能难以使用并且容易出错。identityignore 修饰符可以将语法简化为:

```
db2 import from import.del of del modified by identityignore
   replace into table1
```

当带有标识列的表导出至 IXF 文件时，可使用 IMPORT 命令的 REPLACE_CREATE 和 CREATE 选项来重新创建该表，包括其标识列属性。如果这种 IXF 文件是从包含类型为 GENERATED ALWAYS 的标识列的表创建的，那么只有通过指定 identityignore 修饰符才能成功导入数据文件。否则会拒绝所有行（SQL3550W）。

注: 不推荐使用 IMPORT 命令的 CREATE 和 REPLACE_CREATE 选项，将来的发行版中可能会除去这两个选项。

生成列导入注意事项

无论输入数据是否具有生成列值，都可以使用 import 实用程序将数据导入到包含（非标识）生成列的表中。

如果未使用任何与生成列相关的文件类型修饰符，那么 import 实用程序会遵循下列规则来工作:

- 对于一个生成列，如果输入文件中的相应行缺少该列的值或者显式指定了 NULL 值，那么会为该生成列生成一个值。如果为生成列提供了非空值，那么将拒绝该行（SQL3550W）。
- 如果服务器为不可空生成列生成了 NULL 值，那么会拒绝此字段所属的数据行（SQL0407N）。例如，如果将不可空生成列定义为两个表列之和，但在输入文件中为这两列提供了 NULL 值，那么会发生这种情况。

有两种方法可用来简化将数据导入到包含生成列的表中的操作: generatedmissing 和 generatedignore 文件类型修饰符。

在没有生成列的情况下导入数据

如果输入数据文件不包含表中的所有生成列的任何值（甚至未包含 NULL 值），那么 generatedmissing 修饰符会使您能够更方便地将数据导入到包含生成列的表中。例如，考虑使用以下 SQL 语句定义的表:

```
create table table1 (c1 int,
                     c2 int,
                     g1 int generated always as (c1 + c2),
                     g2 int generated always as (2 * c1),
                     c3 char(1))
```

用户可能想将数据从文件（load.del）导入到 TABLE1 中，此数据可能是从没有任何生成列的表中导出的。下面是此类文件的一个示例:

```
1, 5, J
2, 6, K
3, 7, I
```

导入此文件的一种方法是通过 IMPORT 命令显式列示所要导入的列，如下所示:

```
db2 import from import.del of del replace into table1 (c1, c2, c3)
```

但是，对于包含许多列的表来说，此语法难以使用并且容易出错。另一种导入此文件的方法是使用 generatedmissing 文件类型修饰符，如下所示:

```
db2 import from import.del of del modified by generatedmissing
   replace into table1
```

在具有生成列的情况下导入数据

generatedignore 修饰符在某些方面与 generatedmissing 相反。它向 import 实用程序指示: 即使输入数据文件包含所有生成列的数据，也应该忽略该数据，并且应该为每一行生成值。例如，用户可能想将以下数据按照上述定义从文件（import.del）导入到 TABLE1 中:

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

用户提供的非空值 10、11 和 12（用于 g1）以及 15、16 和 17（用于 g2）导致拒绝该行（SQL3550W）。为了避免这种情况，用户可以发出以下 IMPORT 命令:

```
db2 import from import.del of del method P(1, 2, 5)
   replace into table1 (c1, c2, c3)
```

同样，如果该表包含许多列，那么此方法可能难以使用并且容易出错。generatedignore 修饰符可以将语法简化为:

```
db2 import from import.del of del modified by generatedignore
   replace into table1
```

对于 INSERT_UPDATE，如果生成列同时充当主键并且指定了 generatedignore 修饰符，那么 IMPORT 命令会采用 generatedignore 修饰符。IMPORT 命令不会在 UPDATE 的 WHERE 子句中用用户提供的值替换此列。

LOB 导入注意事项

因为 import 实用程序将单个列值的大小限制为 32 KB，所以导入 LOB 时有一些额外注意事项。

缺省情况下，import 实用程序将输入文件中的数据视为要装入列中的数据。但是，如果大对象（LOB）数据存储在主要输入数据文件中，那么数据大小被限制为 32 KB。因此，为避免丢失数据，应将 LOB 数据存储在主要数据文件以外的位置，并且应在导入 LOB 时指定 lobsinfile 文件类型修饰符。

LOBS FROM 子句会隐式激活 lobsinfile。导入数据时，LOBS FROM 子句会将导入数据时用于搜索 LOB 文件的路径列表传递至 import 实用程序。如果未指定 LOBS FROM 选项，那么假定要导入的 LOB 文件与输入关系数据文件位于同一路径中。

指示存储 LOB 数据的位置

导入 LOB 信息时，可以使用 LOB 位置说明符（LLS）将多个 LOB 存储在单个文件中。指定 lobsinfile 后，export 实用程序会生成 LLS 并将其存储在导出输出文件中，并且 LLS 会指示 LOB 数据的位置。导入使用指定的 lobsinfile 选项修饰的数据时，数据库要求每个对应 LOB 列都有对应的 LLS。如果 LOB 列遇到的不是 LLS，那么数据库会将其视为 LOB 文件，并且将把整个文件作为 LOB 装入。

对于 CREATE 方式下的导入，可通过使用 LONG IN 子句指定将创建 LOB 数据并将其存储在单独表空间中。

以下示例显示如何导入 LOB 存储在不同文件中的 DEL 文件:

```
IMPORT FROM inputfile.del OF DEL
 LOBS FROM /tmp/data
 MODIFIED BY lobsinfile
 INSERT INTO newtable
```

用户定义的单值类型导入注意事项

import 实用程序自动将用户定义的单值类型（UDT）转换为类似的基本数据类型。这样您就不必将 UDT 显式转换为基本数据类型。数据类型转换允许在 UDT 与 SQL 中的基本数据类型之间进行比较。

其他导入注意事项

客户机/服务器环境和导入

将文件导入至远程数据库时，可调用存储过程以对服务器执行导入操作。

在下列情况下不能调用存储过程:

- 应用程序和数据库代码页不同。
- 要导入的文件是包含多个部分的 PC/IXF 文件。
- 用于导入数据的方法是列名或相对列位置。
- 提供的目标列列表的长度超过 4 KB。

- 指定了 LOBS FROM 子句或 lobsinfile 修饰符。
- 对 ASC 文件指定了 NULL INDICATORS 子句。

导入使用存储过程时，将使用服务器上安装的缺省语言在消息文件中创建消息。如果客户机和服务器上的语言相同，那么这些消息使用应用程序的语言。

import 实用程序在 sqllib 目录（或 **DB2INSTPROF** 注册表变量指示的目录，如果指定了的话）的 tmp 子目录中创建两个临时文件。一个文件用于数据，一个文件用于 import 实用程序生成的消息。

如果接收到有关在服务器上写入或打开数据的错误，那么请确保:

- 该目录已存在。
- 有足够的磁盘空间来存储文件。
- 实例所有者在目录中具有写许可权。

导入期间进行表锁定

import 实用程序支持两种表锁定方式: 脱机或 ALLOW NO ACCESS 方式; 以及联机或 ALLOW WRITE ACCESS 方式。

ALLOW NO ACCESS 方式会阻止并行应用程序访问表数据。ALLOW WRITE ACCESS 方式允许并行应用程序同时对导入目标表进行读写访问。如果未显式指定任何方式，那么导入会以缺省方式 ALLOW NO ACCESS 运行。同时，在缺省情况下，import 实用程序会使用隔离级别 RS（读稳定性）绑定至数据库。

脱机导入（ALLOW NO ACCESS）

在 ALLOW NO ACCESS 方式下，导入会在插入任何行之前获取针对目标表的独占（X）锁定。挂起对该表的锁定有两种影响:

- 首先，如果其他应用程序挂起针对导入目标表的表锁定或行锁定，那么 import 实用程序将等待这些应用程序落实或回滚更改。
- 其次，import 实用程序运行时，请求锁定的任何其他应用程序会等待导入操作完成。

注: 可指定锁定超时值，这会避免应用程序（包括 import 实用程序）无限期等待锁定。通过在操作开始时请求互斥锁定，导入可阻止因为其他应用程序正在处理并挂起针对同一目标表的行锁定而导致的死锁。

联机导入（ALLOW WRITE ACCESS）

在 ALLOW WRITE ACCESS 方式下，import 实用程序将获取针对目标表的非独占（IX）锁定。挂起对该表的此锁定具有下列影响:

- 如果其他应用程序挂起不兼容的表锁定，那么在所有这些应用程序落实或回滚更改之前，import 实用程序不会开始插入数据。
- import 实用程序运行时，如果任何其他应用程序请求不可兼容的表锁定，那么这些应用程序都将等待直至导入操作落实或回滚当前事务。注意，导入的表锁定仅对单个事务有效。因此，在每次落实后，联机导入必须请求表锁定并可能需要等待。
- 如果其他应用程序挂起不兼容的行锁定，那么 import 实用程序将停止插入数据直到所有这些应用程序落实或回滚更改。

- import 实用程序运行时，如果任何其他应用程序请求不可兼容的行锁定，那么这些应用程序都将等待直至导入操作落实或回滚当前事务。

为保留联机属性并降低死锁机率，ALLOW WRITE ACCESS 导入将定期落实当前事务，并在上升为独占表锁定之前释放所有行锁定。如果未显式设置落实频率，那么导入会按指定了 COMMITCOUNT AUTOMATIC 的方式落实。如果 COMMITCOUNT 设置为 0，那么不会执行任何落实。

ALLOW WRITE ACCESS 方式与下列各项不兼容:

- 以 REPLACE、CREATE 或 REPLACE_CREATE 方式导入
- 使用缓冲插入导入
- 导入到目标视图中
- 导入到层次结构表中
- 导入到锁定详细程度设置为表级别（通过使用 ALTER TABLE 语句的 LOCKSIZE 参数设置）的表中

参考 - 导入

IMPORT

将具有受支持文件格式的外部文件中的数据插入表、层次结构、视图或昵称中。另一种速度更快的方法是使用 LOAD 实用程序，但是 LOAD 实用程序不支持在层次结构级别装入数据。

指向 第 71 页的『IMPORT 实用程序的文件类型修饰符』的快速链接。

权限

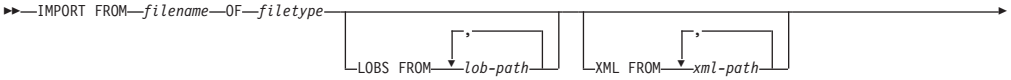
- 使用 INSERT 选项进行 IMPORT 时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对参与的每个表、视图或昵称的 CONTROL 特权
  - 对参与的每个表或视图的 INSERT 和 SELECT 特权
- 使用 INSERT_UPDATE 选项 IMPORT（导入）到现有表时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对参与的每个表、视图或昵称的 CONTROL 特权
  - 对参与的每个表或视图的 INSERT、SELECT、UPDATE 和 DELETE 特权
- 使用 REPLACE 或 REPLACE_CREATE 选项 IMPORT（导入）到现有表时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对表或视图的 CONTROL 特权
  - 对表或视图的 INSERT、SELECT 和 DELETE 特权

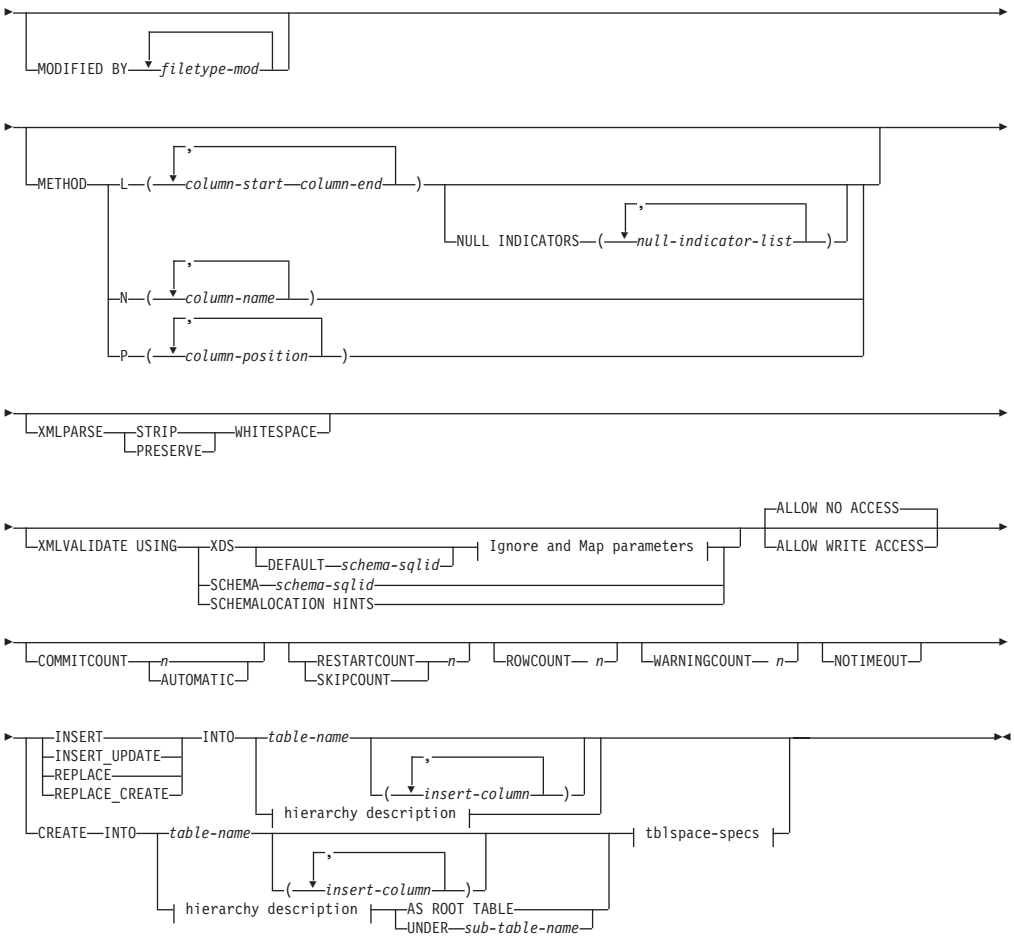
- 使用 CREATE 或 REPLACE_CREATE 选项 IMPORT（导入）到新表时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对数据库的 CREATETAB 权限、对表空间的 USE 特权以及下列其中一项权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果该表的隐式或显式模式名称不存在的话）
    - 对模式的 CREATEIN 特权（如果该表的模式名称引用现有模式的话）
- 使用 CREATE 或 REPLACE_CREATE 选项 IMPORT（导入）到不存在的层次结构时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对数据库的 CREATETAB 权限、对表空间的 USE 特权以及下列其中一项权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果表的模式名称不存在的话）
    - 对模式的 CREATEIN 特权（如果表的模式存在的话）
    - 对层次结构中的每个子表的 CONTROL 特权（如果使用了整个层次结构上的 REPLACE_CREATE 选项的话）
- 使用 REPLACE 选项 IMPORT（导入）到现有层次结构时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对层次结构中每个子表的 CONTROL 特权
- 要将数据导入到包含受保护列的表中，会话授权标识必须拥有允许对该表中所有受保护列执行写访问的 LBAC 凭证。否则，导入将失败并且返回错误（SQLSTATE 42512）。
- 要将数据导入到包含受保护行的表中，会话授权标识必须拥有满足下列条件的 LBAC 凭证:
  - 它是用于保护表的安全策略的一部分
  - 已将它授予会话授权标识以进行写访问

  要插入的行上的标号、用户的 LBAC 凭证、安全策略定义和 LBAC 规则共同确定行上的标号。
- 如果指定了 REPLACE 或 REPLACE_CREATE 选项，那么会话授权标识必须有权删除该表。

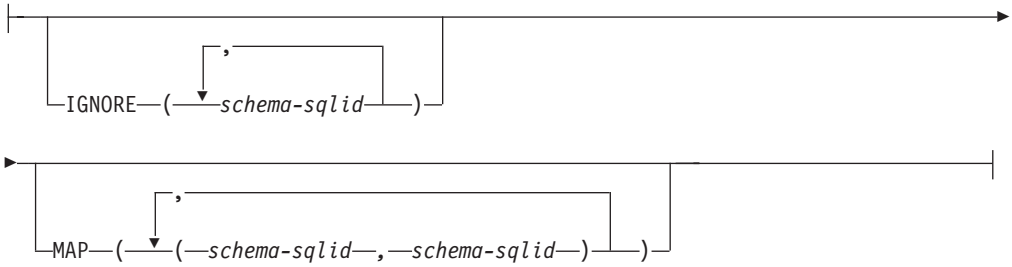
必需的连接

命令语法



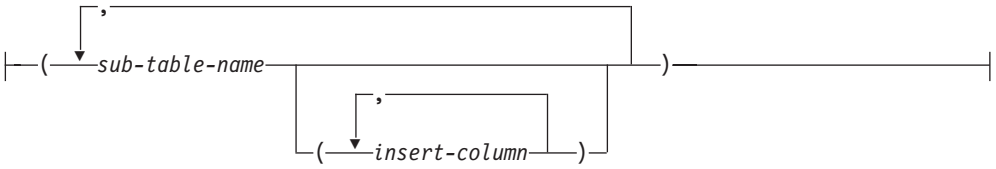


Ignore and Map parameters:



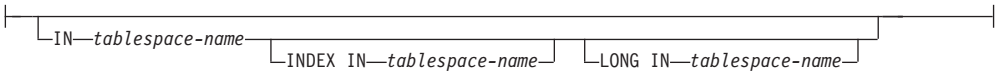
hierarchy description:

sub-table-list:



traversal-order-list:

tblspace-specs:



命令参数

ALL TABLES

这是仅用于层次结构的隐式关键字。当导入层次结构时，缺省情况是导入按遍历顺序指定的所有表。

ALLOW NO ACCESS

以脱机方式运行导入操作。在插入任何行前，就会获得对目标表的互斥（X）锁定。这将阻止并发应用程序访问表数据。这是缺省导入行为。

ALLOW WRITE ACCESS

以联机方式运行导入操作。插入第一行时，将获得对目标表的意向互斥（IX）锁定。这允许并发阅读器和写程序访问表数据。联机方式与 REPLACE、CREATE 或 REPLACE_CREATE 导入选项不兼容。不支持将联机方式与缓冲插入一起使用。导入操作将定期落实已插入的数据，以防止锁定升级为表锁定，并且避免用尽活动日志空间。即使未使用 COMMITCOUNT 选项，也将执行这些落实操作。在每次落实时，导入操作都将失去它的 IX 表锁定，但是在落实后又将尝试重新获取锁定。当导入到昵称时，此参数是必需的，并且必须为 COMMITCOUNT 指定一个有效数（AUTOMATIC 不被视为有效选项）。

AS ROOT TABLE

创建一个或多个子表作为独立表层次结构。

COMMITCOUNT *n* | AUTOMATIC

导入 *n* 条记录后，将执行 COMMIT。当指定了数目 *n* 时，IMPORT 实用程序在每导入 *n* 条记录后就会执行 COMMIT。使用复合插入时，会将用户指定的落实频率 *n* 向上舍入为复合计数值的第一个整数倍数。当指定了 AUTOMATIC 时，IMPORT 实用程序将在内部确定需要执行落实的时间。该实用程序将因下列任一原因而进行落实:

- 为了避免用尽活动日志空间
- 为了避免使锁定从行级别升级到表级别

如果指定了 ALLOW WRITE ACCESS 选项，而未指定 COMMITCOUNT 选项，那么 IMPORT 实用程序将执行落实操作，如同指定了 COMMITCOUNT AUTOMATIC 一样。

如果在插入或更新记录时 IMPORT 命令遇到了 SQL0964C（事务日志已满）并且指定了 COMMITCOUNT *n*，那么 IMPORT 将尝试通过执行无条件的落实，然后重新尝试插入或更新记录来解决此问题。如果这样做不能帮助您解决日志已满的情况（也就是因对数据库执行其他活动而使日志已满的情况），那么 IMPORT 命令将像预期那样失败，但是落实的行数可能不是 COMMITCOUNT *n* 值的倍数。可以使用 RESTARTCOUNT 或 SKIPCOUNT选项来避免处理已经落实的那些行。

CREATE

注: 不推荐使用 CREATE 参数，将来的发行版中可能会将其除去。有关其他详细信息，请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”。

采用数据库的代码页来创建表定义和行内容。如果数据是从 DB2 表、子表或层次结构中导出的，那么将创建索引。如果将此选项应用于层次结构，并且数据是从 DB2 中导出的，那么还将创建类型层次结构。此选项只能用于 IXF 文件。

当导入到昵称时，此参数无效。

注: 如果数据是从 MVS™ 主机数据库中导出的，并且它包含一些 LONGVAR 字段，而按照页大小计算的这些字段的长度超过了 254，那么 CREATE 操作可能会因行太长而失败。请参阅『重新创建导入的表』，以获取限制列表。在这种情况下，应该手动创建表，并且应该调用指定了 INSERT 的 IMPORT，或者应该使用 LOAD 命令。

DEFAULT *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。通过 DEFAULT 子句指定的模式标识: 当已导入的 XML 文档的 XML 数据说明符（XDS）不包含用于标识 XML 模式的 SCH 属性时要用于验证的模式。

DEFAULT 子句优先于 IGNORE 和 MAP 子句。如果 XDS 满足 DEFAULT 子句，那么将忽略 IGNORE 和 MAP 规范。

FROM *filename*

HIERARCHY

指定要导入分层数据。

IGNORE *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。如果 SCH 属性标识了一种或多种模式，那么 IGNORE 子句指定这些模式中要忽略的模式列表。如果 SCH 属性存在于已导入的 XML 文档的 XML 数据说明符中，并且由 SCH 属性标识的模式包含在要忽略的模式列表中，那么将不会对已导入的 XML 文档进行模式验证。

如果某一模式是在 IGNORE 子句中指定的，那么该模式不能存在于 MAP 子句中模式对的左边。

IGNORE 子句仅适用于 XDS。如果 IGNORE 子句指定了由 MAP 子句映射的模式，那么以后不会忽略该模式。

IN *tablespace-name*

确定将在其中创建表的表空间。该表空间必须存在，并且必须是 REGULAR（常规）表空间。如果未指定其他表空间，那么所有表部件都将存储在此表空间中。如果未指定此子句，那么将在由授权标识创建的表空间中创建表。如果找不到表空间，那么会将该表放入缺省表空间 USERSPACE1 中。如果已经删除了 USERSPACE1，那么表创建操作将失败。

INDEX IN *tablespace-name*

确定将在其中对表创建索引的表空间。仅当在 IN 子句中指定的主表空间是一个 DMS 表空间时，才允许使用此选项。指定的表空间必须存在，并且必须是 REGULAR 或 LARGE DMS 表空间。

注: 仅当创建了表时，才能指定哪个表空间将包含索引。

insert-column

指定要将数据插入到的表或视图中的列名。

INSERT

将已导入的数据添加至表，但不更改现有表数据。

INSERT_UPDATE

将已导入的数据行添加至目标表，或者使用相匹配的主键来更新（目标表的）现有行。

INTO *table-name*

指定要将数据导入到的数据库表。此表不能是系统表、已声明临时表或者总结表。

应该使用标准表名或者非限定表名时，除了使用早期版本的服务器的情况以外，可以对 INSERT、INSERT_UPDATE 或 REPLACE 使用别名。标准表名的格式为: *schema.tablename*。*schema* 是创建表时所使用的用户名。

LOBS FROM *lob-path*

LOB 数据文件的名称存储在将装入到 LOB 列的那列的主数据文件（ASC、DEL 或 IXF）中。最多可指定 999 个路径。这将隐式激活 LOBSINFILE 行为。

当导入到昵称时，此参数无效。

LONG IN *tablespace-name*

确定将用来存储任何长列（LONG VARCHAR、LONG VARGRAPHIC、LOB 数据类型或者将任何这些数据类型作为源类型的单值类型）的值的表空间。仅当在 IN 子句中指定的主表空间是一个 DMS 表空间时，才允许使用此选项。该表空间必须存在，并且必须是 LARGE DMS 表空间。

MAP *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。使用 MAP 子句来指定要使用的替代模式，这些模式将取代由已导入的每个 XML 文档的 XML数据说明符（XDS）的 SCH 属性所指定的那些模式。MAP 子句指定一个或多个模式对的列表，而每个模式对都表示一个模式与另一个模式之间的映射。模式对中的第一个模式表示 XDS 中的 SCH 属性引用的模式。模式对中的第二个模式表示应该用来执行模式验证的模式。

如果某一模式存在于 MAP 子句中的模式对的左边，那么不能在 IGNORE 子句中也指定该模式。

一旦应用了模式对映射，其结果就是最终结果。映射操作不是过渡操作，因此，以后不会将所选择的模式应用于另一个模式对映射。

不能多次映射同一模式，这就意味着该模式不会多次出现在一个模式对的左边。

METHOD

L 指定要将数据导入到的列的起始列号和结束列号。列号就是与一个数据行的开头的字节偏移量。它是从 1 开始计数的。

注: 此方法只能用于 ASC 文件，并且对于该文件类型是唯一有效的选项。

N 指定要导入的数据文件中的列名。这些列名的大小写必须与系统目录中相应名称的大小写相匹配。不可空的每个表列在 METHOD N 列表中都应该具有相应的条目。例如，给定数据字段 F1、F2、F3、F4、F5 和 F6，以及表列 C1 INT、C2 INT NOT NULL、C3 INT NOT NULL 和 C4 INT，那么 method N (F2, F1, F4, F3) 是有效请求，而 method N (F2, F1) 是无效请求。

注: 此方法只能用于 IXF 文件。

P 指定要导入的输入数据字段的字段编号。

注: 此方法只能用于 IXF 或 DEL 文件，并且对于 DEL 文件类型是唯一有效的选项。

MODIFIED BY *filetype-mod*

指定文件类型修饰符选项。请参阅 第 71 页的『IMPORT 实用程序的文件类型修饰符』。

NOTIMEOUT

指定 IMPORT 实用程序在等待锁定时不会超时。此选项取代了数据库配置参数 *locktimeout*。其他应用程序将不受影响。

NULL INDICATORS *null-indicator-list*

仅当指定了 METHOD L 参数时，才能使用此选项。即，输入文件是 ASC 文件。空指示符列表是用于指定每个空指示符字段的正整数的列表，各个正整数之间用逗号隔开。列号是空指示符字段与一个数据行的开头之间的字节偏移量。对于 METHOD L 参数中定义的每个数据字段，在空指示符列表中必须有一个其对应的条目。如果列号为 0，那么表示相应的数据字段中始终包含数据。

如果空指示符列中的值为 `Y`，那么表示列数据为 NULL。如果空指示符列中的值是除了 `Y` 之外的任何字符，那么表示列数据不是 NULL，并且将导入由 METHOD L 选项指定的列数据。

可通过将 MODIFIED BY 选项与文件类型修饰符 `nullindchar` 一起使用来更改空指示符。

OF *filetype*

指定输入文件中数据的格式:

- ASC（非定界 ASCII 格式）
- DEL（定界 ASCII 格式），多种数据库管理器和文件管理器程序使用此格式
- WSF（工作表格式），下列程序使用此格式:

- Lotus 1-2-3
- Lotus Symphony

• IXF（PC 版本的集成交换格式）是一种专门供 DB2 使用的二进制格式。

当导入到昵称时，WSF 文件类型不受支持。

REPLACE

通过截断数据对象来删除表中现有的所有数据，然后插入已导入的数据。表定义和索引定义不会发生更改。仅当表存在时才能使用此选项。如果在层次结构之间移动数据时使用了此选项，那么只能替换整个层次结构的数据，而不能替换单个子表的数据。

当导入到昵称时，此参数无效。

此选项不支持 CREATE TABLE 语句的 NOT LOGGED INITIALLY (NLI) 子句或者 ALTER TABLE 语句的 ACTIVE NOT LOGGED INITIALLY 子句。

如果包含 REPLACE 选项的导入操作与调用了 NLI 子句的 CREATE TABLE 或 ALTER TABLE 语句在同一事务内执行，那么此导入操作不能识别该 NLI 子句。将记录所有插入操作。

变通方法 1

使用 DELETE 语句删除表的内容，然后使用 INSERT 语句调用导入操作

变通方法 2

删除表后再重新创建表，然后使用 INSERT 语句调用导入操作。

DB2 通用数据库™版本 7 和 DB2 UDB 版本 8 存在此局限性

REPLACE_CREATE

注: 不推荐使用 REPLACE_CREATE 参数，将来的发行版中可能会将其除去。有关其他详细信息，请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”。

如果表存在，那么通过截断数据对象来删除表中现有的所有数据，然后插入已导入的数据，但不更改表定义或索引定义。

如果表不存在，那么采用数据库的代码页创建表和索引定义以及行内容。请参阅*重新创建导入的表*，以获取限制列表。

此选项只能用于 IXF 文件。如果在层次结构之间移动数据时使用了此选项，那么只能替换整个层次结构的数据，而不能替换单个子表的数据。

当导入到昵称时，此参数无效。

RESTARTCOUNT *n*

指定要从第 *n* + 1 条记录开始执行导入操作。将跳过前 *n* 条记录。此选项的功能相当于 SKIPCOUNT。RESTARTCOUNT 和 SKIPCOUNT 是互斥的。

ROWCOUNT *n*

指定要导入（插入或更新）文件中的 *n* 条物理记录。允许用户从文件中仅导入 *n* 行（从由 SKIPCOUNT 或 RESTARTCOUNT 选项确定的记录开始算起）。如果未指定 SKIPCOUNT 或 RESTARTCOUNT 选项，那么会导入前 *n* 行。如果指定了 SKIPCOUNT *m* 或 RESTARTCOUNT *m*，那么将导入从第 *m*+1 行到第 *m*+*n* 行。使用复合插入时，会将用户指定的 ROWCOUNT *n* 向上舍入为复合计数值的第一个整数倍数。

SKIPCOUNT *n*

指定要从第 *n* + 1 条记录开始执行导入操作。将跳过前 *n* 条记录。此选项的功能相当于 RESTARTCOUNT。SKIPCOUNT 和 RESTARTCOUNT 是互斥的。

STARTING *sub-table-name*

这是仅用于层次结构的一个关键字，请求缺省顺序，从 *sub-table-name* 开始。对于 PC/IXF 文件，缺省顺序是存储在输入文件中的顺序。对于 PC/IXF 文件格式，缺省顺序是唯一有效的顺序。

sub-table-list

对于使用 INSERT 或 INSERT_UPDATE 选项的类型表，使用子表名的列表来指示要将数据导入到的子表。

traversal-order-list

对于使用 INSERT、INSERT_UPDATE 或 REPLACE 选项的类型表，使用子表名的列表来指示导入层次结构中的子表的遍历顺序。

UNDER *sub-table-name*

指定用于创建一个或多个子表的父表。

WARNINGCOUNT *n*

在发出 *n* 个警告后停止导入操作。如果希望不产生警告，那么设置此参数，但是需要验证使用的是正确的文件和表。如果指定了不正确的导入文件或目标表，那么 IMPORT 实用程序将对它试图导入的每一行生成一个警告，这将导致导入失败。如果 *n* 为 0，或者未指定此选项，那么无论发出多少个警告，都将继续执行导入操作。

XML FROM *xml-path*

指定一个或多个包含 XML 文件的路径。

XMLPARSE

指定如何解析 XML 文档。如果未指定此选项，那么将由 CURRENT XMLPARSE OPTION 专用寄存器的值来确定 XML 文档的解析行为。

STRIP WHITESPACE

指定在解析 XML 文档时要去掉空格。

PRESERVE WHITESPACE

指定在解析 XML 文档时不去掉空格。

XMLVALIDATE

指定在适当的情况下将针对某一模式来验证 XML 文档。

USING XDS

将针对由主数据文件中的 XML 数据说明符（XDS）标识的 XML 模式来验证 XML 文档。缺省情况下，如果使用 USING XDS 子句调用了 XMLVALIDATE 选项，那么将由 XDS 的 SCH 属性来确定用来执行验证的模式。如果 XDS 中不存在 SCH 属性，那么除非缺省模式是由 DEFAULT 子句指定的，否则将不会进行模式验证。

可以使用 DEFAULT、IGNORE 和 MAP 子句来修改模式确定行为。这三个可选子句直接应用于 XDS 的指定，但是它们不会互相应用。例如，如果由于 DEFAULT 子句指定了某一模式而选择了该模式，那么，即使 IGNORE 子句也指定了该模式，该模式仍不会被忽略。类似，如

果由于将某一模式指定为 MAP 子句对的第一部分而选择了该模式，那么，即使在另一个 MAP 子句对的第二部分中也指定了该模式，该模式仍不会被重新映射。

USING SCHEMA *schema-sqlid*
将针对具有指定的 SQL 标识的 XML 模式来验证 XML 文档。在这种情况下，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

USING SCHEMALOCATION HINTS
将针对由源 XML 文档中的 XML 模式位置提示标识的模式来验证 XML 文档。如果在 XML 文档中找不到 schemaLocation 属性，那么将不执行验证。指定 USING SCHEMALOCATION HINTS 子句时，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

请参阅下面的 XMLVALIDATE 选项示例。

使用说明

在开始执行导入操作前，务必完成所有表操作并释放所有锁定。这可以通过在关闭使用 WITH HOLD 打开的所有游标前发出 COMMIT，或者通过发出 ROLLBACK 来完成。

IMPORT 实用程序使用 SQL INSERT 语句将一些行添加至目标表。该实用程序将对输入文件中的每行数据发出一个 INSERT 语句。如果 INSERT 语句失败，那么将导致执行以下两个操作之一:

- 如果后续 INSERT 语句可能会成功，那么会将警告消息写入消息文件中，但是将继续进行处理。
- 如果后续 INSERT 语句可能会失败，并且数据库可能会被破坏，那么会将错误消息写入消息文件并停止处理。

在执行 REPLACE 或 REPLACE_CREATE 操作期间删除了旧的行后，该实用程序将执行自动落实（COMMIT）。因此，如果系统失败，或者在截断表对象后应用程序中断了数据库管理器，那么所有旧数据都会丢失。确保在使用这些选项前不再需要旧数据。

如果在执行 CREATE、REPLACE 或 REPLACE_CREATE 操作期间日志已满，那么该实用程序将对已插入的记录执行自动落实（COMMIT）。如果系统失败，或者在自动落实（COMMIT）后应用程序中断了数据库管理器，那么具有初始数据的表将保留在数据库中。使用 REPLACE 或 REPLACE_CREATE 选项来重新运行整个导入操作，或者在 RESTARTCOUNT 参数设置为已成功导入的行数的情况下执行 INSERT 操作。

缺省情况下，不会对 INSERT 或 INSERT_UPDATE 选项执行自动落实（COMMIT）。但是，如果 COMMITCOUNT 参数不为零，那么将执行自动落实（COMMIT）。如果未执行自动落实（COMMIT），当日志已满时就会导致回滚（ROLLBACK）。

如果存在下列任何一种情况，脱机导入将不执行自动落实（COMMIT）:

- 目标是视图而不是表
- 使用了复合插入
- 使用了缓冲插入

缺省情况下,联机导入时将执行自动落实（COMMIT），以同时释放活动日志空间和锁定列表。仅当将 COMMITCOUNT 值指定为 0 时，才不会执行自动落实（COMMIT）。

每当 IMPORT 实用程序执行落实（COMMIT）时，会将两条消息写入消息文件中：一条消息指示要落实的记录数；在成功执行落实（COMMIT）后将写入另一条消息。当在失败后重新启动导入操作时，请指定要跳过的记录数，此数目是在上一次成功执行落实（COMMIT）时确定的。

IMPORT 实用程序将接受带有次要的不兼容问题的输入数据（例如，可通过使用填充或截断来导入的字符数据，以及可以使用另一种数字数据类型来导入的数字数据），但是不会接受具有主要的不兼容问题的数据。

如果一个对象表具有除它本身之外的任何从属项，或者如果对象视图的基本表具有任何从属项（包括它本身），那么不能对该对象表或对象视图执行 REPLACE 或 REPLACE_CREATE。要替换这样的一个表或视图，请执行以下操作:

1. 删除该表是父表的所有外键。
2. 运行 IMPORT 实用程序。
3. 改变该表以重新创建外键。

如果在重新创建外键时发生了错误，那么修改数据以保持引用完整性。

根据 PC/IXF 文件重新创建表时，将不保留引用约束和外键定义。如果数据先前是使用 SELECT * 导出的，那么会保留主键定义。

在导入到远程数据库时，要求服务器上有足够的磁盘空间来复制输入数据文件、输出消息文件以及满足数据库可能增大的需求。

如果对远程数据库运行导入操作，并且输出消息文件很长（超过 60 KB），那么返回给客户机上的用户的消息文件可能会在导入操作过程中丢失消息。但始终会保留前 30 KB 的消息信息和最后 30 KB 的消息信息。

如果 PC/IXF 文件位于硬盘驱动器而不是软盘上，那么将 PC/IXF 文件导入到远程数据库中时速度会更快。

必须存在数据库表或层次结构，才能导入采用 **ASC**、**DEL** 或 **WSF** 文件格式的数据；但是，如果表尚不存在，那么 IMPORT CREATE 或 IMPORT REPLACE_CREATE 在从 PC/IXF 文件中导入数据时将创建表。对于类型表，IMPORT CREATE 可以创建类型层次结构以及表层次结构。

应该使用 PC/IXF 导入来在数据库之间移动数据（包括分层数据）。如果包含行分隔符的字符数据已导出到定界 ASCII（DEL）文件并且已由文本传输程序处理，那么包含行分隔符的字段将收缩或展开。如果从同一台客户机可以访问源数据库和目标数据库，那么不需要执行文件复制步骤。

默认 ASC 和 DEL 文件中的数据采用的是执行导入操作的客户机应用程序的代码页。当导入采用其他代码页的数据时，建议使用 PC/IXF 文件（这些文件允许使用其他代码页）。如果 PC/IXF 文件与 IMPORT 实用程序采用相同的代码页，那么像常规应用程序一样进行处理。如果它们使用不同的代码页，并且指定了 FORCEIN 选项，那么 IMPORT 实用程序将认为 PC/IXF 文件中的数据与执行导入操作的应用程序具有相同的代码页。即使这两种代码页之间存在转换表，也会发生这种情况。如果它们使用不同

的代码页，并且未指定 FORCEIN 选项，代码页之间还存在转换表，那么 PC/IXF 文件中的所有数据都将从文件代码页转换为应用程序代码页。如果它们使用不同的代码页，并且未指定 FORCEIN 选项，代码页之间不存在转换表，那么导入操作将失败。这仅适用于采用 AIX 操作系统的 DB2 客户机上的 PC/IXF 文件。

对于 8 KB 页（接近 1012 列这一限制）上的表对象，导入 PC/IXF 数据文件可能会导致 DB2 返回错误，这是因为已经超过了 SQL 语句的最大大小。仅当列类型为 CHAR、VARCHAR 或 CLOB 时才会发生这种情况。该限制并不适用于导入 **DEL** 或 **ASC** 文件这种情况。如果使用 PC/IXF 文件来创建新表，那么替代方法是使用 db2look 来转储用于创建该表的 DDL 语句，然后通过 CLP 发出该语句。

可以使用 DB2 Connect 将数据导入到 DRDA 服务器（例如，DB2 OS/390 版、DB2 VM 和 VSE 版以及 DB2 OS/400 版）中。仅支持 PC/IXF 导入（INSERT 选项）。还支持 RESTARTCOUNT 参数，但不支持 COMMITCOUNT 参数。

对类型表使用 CREATE 选项时，将创建在 PC/IXF 文件中定义的每个子表；不能改变子表定义。对类型表使用除 CREATE 之外的其他选项时，遍历顺序列表允许用户指定遍历顺序；因此，遍历顺序列表必须与在执行导出操作期间所使用的遍历顺序列表相匹配。对于 PC/IXF 文件格式，用户只需要指定目标子表名，并使用文件中存储的遍历顺序。

可以使用 IMPORT 实用程序来恢复先前已导出到 PC/IXF 文件的表。该表将返回到导出它时所处的状态。

不能将数据导入到系统表、已声明临时表或总结表。

不能通过 import 实用程序创建视图。

支持导入多部件 PC/IXF 文件，该文件的各个部件是从 Windows 系统复制到 AIX 系统的。仅必须在 IMPORT 命令中指定第一个文件的名称。例如，`IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1`。文件 data.002 等应该位于 `data.ixf` 所在的目录中。

在 Windows 操作系统上:

- 不支持导入按逻辑划分的 PC/IXF 文件。
- 不支持导入错误格式的 PC/IXF 或 WSF 文件。

采用内部格式的安全标号可能包含换行符。如果导入使用 DEL 文件格式的文件，那么这些换行符可能会被误认为定界符。如果发生此问题，请通过在 IMPORT 命令中指定文件类型修饰符 `delprioritychar` 来对定界符使用较旧的缺省优先级。

联合注意事项

使用 IMPORT 命令和 INSERT、UPDATE 或 INSERT_UPDATE 命令参数时，必须确保您对参与的昵称具有 CONTROL 特权。必须确保您在执行导入操作时想使用的昵称已存在。还存在一些应注意的限制，在 IMPORT 命令参数部分说明了这些限制。

某些数据源（例如 ODBC）不支持导入到昵称。

IMPORT 实用程序的文件类型修饰符

表 14. IMPORT 实用程序的有效文件类型修饰符: 所有文件格式

修饰符	描述
compound=*x*	*x* 是一个 1 到 100 之间（包括 1 和 100 在内）的数字。使用无原子的复合 SQL 来插入数据，并且每次都将尝试使用 *x* 语句。 如果指定了此修饰符，而事务日志不是足够大，那么导入操作将失败。事务日志必须足够大，用来容纳由 COMMITCOUNT 指定的行数；如果未指定 COMMITCOUNT，那么用来容纳数据文件中的行数。因此，建议指定 COMMITCOUNT 选项以避免事务日志溢出。 此修饰符与 INSERT_UPDATE 方式、分层表和下列修饰符不兼容: `usedefaults`、`identitymissing`、`identityignore`、`generatedmissing` 和 `generatedignore`。
generatedignore	此修饰符通知 IMPORT 实用程序: 所有生成列的数据在数据文件中都存在，但是应该忽略这些数据。这将导致所有生成列值都由该实用程序生成。此修饰符不能与 `generatedmissing` 修饰符一起使用。
generatedmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含生成列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。此修饰符不能与 `generatedignore` 修饰符一起使用。
identityignore	此修饰符通知 IMPORT 实用程序: 标识列的数据在数据文件中已存在，但是应该忽略该数据。这将导致所有标识值都由该实用程序生成。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 `identitymissing` 修饰符一起使用。
identitymissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含标识列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。此修饰符不能与 `identityignore` 修饰符一起使用。
lobsinfile	*lob-path* 指定包含 LOB 数据的文件所在的路径。 每个路径都至少包含这样一个文件: 该文件至少包含数据文件中的“Lob 位置说明符”（LLS）所指向的一个 LOB。对于存储在 LOB 文件路径中的文件，LLS 就是对这些文件中的 LOB 所在位置的字符串表示。LLS 的格式为 *filename.ext.nnn.mmm/*，其中 *filename.ext* 是包含 LOB 的文件名，*nnn* 是文件中的 LOB 的偏移量（以字节计），*mmm* 是 LOB 的长度（以字节计）。例如，如果 `db2exp.001.123.456/` 字符串存储在数据文件中，那么 LOB 位于 db2exp.001 文件中偏移量为 123 的位置，其长度为 456 字节。 使用“lobsinfile”修饰符时，LOBS FROM 子句指定 LOB 文件所在的位置。LOBS FROM 子句将隐式激活 LOBSINFILE 行为。导入数据时，LOBS FROM 子句将要从其中搜索 LOB 文件的路径列表传递给 IMPORT 实用程序。 要指示一个空 LOB，应将大小输入为 -1。如果将大小指定为 0，那么会将它视作长度为 0 的 LOB。对于长度为 -1 的 LOBS，将忽略偏移量和文件名。例如，空 LOB 的 LLS 可能是 db2exp.001.7.-1/。
no_type_id	仅当导入到单个子表中时才有效。典型用途是从常规表中导出数据，然后（使用此修饰符）调用导入操作来将该数据转换到单个子表中。

表 14. IMPORT 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
nodefaults	如果未显式地指定目标表列的源列，而表列又不可空，那么不会装入缺省值。在不使用此选项的情况下，如果未显式地指定其中一个目标表列的源列，那么将发生下列情况之一: • 如果可以为一列指定缺省值，那么将装入缺省值 • 如果该列可空，且不能为该列指定缺省值，那么将装入 NULL • 如果该列不可空，但不能为该列指定缺省值，那么将返回错误，并且该实用程序将停止处理。
norowwarnings	抑制关于被拒绝行的所有警告。
rowchangetimestampignore	此修饰符通知 IMPORT 实用程序: ROW CHANGE TIMESTAMP 列的数据在数据文件中已存在，但是应该忽略该数据。这将导致该实用程序生成所有 ROW CHANGE TIMESTAMP。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 `rowchangetimestampmissing` 修饰符一起使用。
rowchangetimestampmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含 ROW CHANGE TIMESTAMP 列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。此修饰符不能与 `rowchangetimestampignore` 修饰符一起使用。
seclabelchar	指示输入源文件中的安全标号采用安全标号值的字符串格式，而不是采用缺省编码数字格式。当装入每个安全标号时，IMPORT 实用程序会将它们转换为内部格式。如果字符串的格式不正确，那么将不装入该行，并且将返回警告（SQLSTATE 01H53）。如果字符串并不表示作为用于保护表的安全策略的一部分的有效安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3243W）。 如果指定了 seclabelname 修饰符，那么不能指定此修饰符，否则导入会失败并且将返回错误（SQLCODE SQL3525N）。
seclabelname	指示输入源文件中的安全标号是由它们的名称指示的，而不是由缺省编码数字格式指示的。如果存在名称，那么 IMPORT 实用程序会将它转换为相应的安全标号。如果不存在具有所指示的（用于保护表的）安全策略名称的安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3244W）。 如果指定了 seclabelchar 修饰符，那么不能指定此修饰符，否则导入会失败并且将返回错误（SQLCODE SQL3525N）。 **注**: 如果文件类型是 ASC，那么安全标号名称后面的任何空格都将被解释为该名称的一部分。为了避免此问题，可以使用文件类型修饰符 striptblanks 来确保除去空格。

表 14. IMPORT 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
usedefaults	如果已经指定了目标表列的源列，但是源列中不包含一个或多个行实例的数据，那么将装入缺省值。缺少的数据的示例如下: • 对于 DEL 文件: 为列值指定了两个相邻的列定界符（“,,”）或者两个相邻的列定界符之间还有任意数目的空格（“, ,”）。 • 对于 DEL/ASC/WSF 文件: 没有足够的列数或者对于原始规范来说不是足够长的行。**注**: 对于 ASC 文件，并不将列值为 NULL 认为是显式丢失，也不会用缺省值来代替 NULL 列值。NULL 列值是由数字、日期、时间和时间戳记列的所有空格字符来表示的，或者是通过对任何类型的列使用 NULL INDICATOR 来指示该列为 NULL 这样来表示的。 在不使用此选项的情况下，如果源列中不包含行实例的数据，那么将发生下列情况之一: • 对于 DEL/ASC/WSF 文件: 如果该列可为空，那么将装入 NULL。如果该列不可空，那么实用程序将拒绝该行。

表 15. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）

修饰符	描述
codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输入数据集中的数据的代码页。在导入操作期间，将采用此代码页的字符数据转换为采用应用程序代码页。 下列规则适用: • 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。 • `nullindchar` 必须指定标准 ASCII 代码集中包含的代码点 x20 与 x7F 之间（包括 x20 和 x7F 在内）的符号。这指的是 ASCII 符号和代码点。 **注**: 1. `codepage` 修饰符不能与 `lobsinfile` 修饰符一起使用。 2. 如果在将代码页从应用程序代码页转换为数据库代码页时进行了数据扩展，那么数据可能会被截断，并且有可能丢失数据。

dateformat="*x*"	*x* 是源文件中的数据所采用的格式[2]。有效日期元素包括:
	`YYYY - 年份（四位数，范围是 0000 到 9999）`
	`M - 月份（一位数或两位数，范围是 1 到 12）`
	`MM - 月份（两位数，范围是 1 到 12; 与 M 元素互斥）`
	`D - 日（一位数或两位数，范围是 1 到 31）`
	`DD - 日（两位数，范围是 1 到 31; 与 D 元素互斥）`
	`DDD - 一年中的某日（三位数，范围是 001 到 366; 与其他日或月份元素互斥）`
	
	对于未指定的每个元素，将为它指定缺省值 1。以下是日期格式的一些示例:
	`"D-M-YYYY"`
	`"MM.DD.YYYY"`
	`"YYYYDDD"`

implieddecimal	隐含的小数点所在的位置由列定义来确定; 不再假定它位于值的末尾。例如，会将值 `12345` 作为 `123.45` 而*不是* `12345.00` 装入 DECIMAL(8,2) 列。

表 15. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
timeformat="*x*"	*x* 是源文件中的时间格式[2]。有效时间元素包括: H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24） HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥） M - 分钟（一位数或两位数，范围是 0 到 59） MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥） S - 秒（一位数或两位数，范围是 0 到 59） SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥） SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥） TT - 正午指示符（AM 或 PM） 对于未指定的每个元素，将为它指定缺省值 0。以下是时间格式的一些示例: "HH:MM:SS" "HH.MM TT" "SSSSS"

表 15. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）(续)

修饰符	描述

timestampformat="*x*"	*x* 是源文件中的时间戳记格式[2]。有效时间戳记元素包括:
	
	YYYY - 年份（四位数，范围是 0000 到 9999）
	M - 月份（一位数或两位数，范围是 1 到 12）
	MM - 月份（两位数，范围是 01 到 12；与 M 和 MMM 元素互斥）
	MMM - 月份（由三个不区分大小写的字母组成的月份名称缩写；与 M 和 MM 元素互斥）
	D - 日（一位数或两位数，范围是 1 到 31）
	DD - 日（两位数，范围是 1 到 31；与 D 元素互斥）
	DDD - 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥）
	H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24）
	HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥）
	M - 分钟（一位数或两位数，范围是 0 到 59）
	MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥）
	S - 秒（一位数或两位数，范围是 0 到 59）
	SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥）
	SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥）
	UUUUUU - 微秒（六位数，范围是 000000 到 999999；与所有其他微秒元素互斥）
	UUUUU - 微秒（五位数，范围是 00000 到 99999，映射至范围 000000 到 999990；与所有其他微秒元素互斥）
	UUUU - 微秒（四位数，范围是 0000 到 9999，映射至范围 000000 到 999900；与所有其他微秒元素互斥）
	UUU - 微秒（三位数，范围是 000 到 999，映射至范围 000000 到 999000；与所有其他微秒元素互斥）
	UU - 微秒（两位数，范围是 00 到 99，映射至范围 000000 到 990000；与所有其他微秒元素互斥）
	U - 微秒（一位数，范围是 0 到 9，映射至范围 000000 到 900000；与所有其他微秒元素互斥）
	TT - 正午指示符（AM 或 PM）
	
	对于未指定的 YYYY、M、MM、D、DD 或 DDD 元素，将为它们指定缺省值 1。对于未指定的 MMM 元素，将为它指定缺省值“Jan”。对于所有其他未指定的元素，将为它们指定缺省值 0。以下是一个表示时间戳记格式的示例:
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 元素的有效值包括: 'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov' 和 'dec'。这些值都不区分大小写。
	
	以下示例说明如何将包含用户定义的日期和时间格式的数据导入到一个称为 schedule 的表中:
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

表 15. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
usegraphiccodepage	如果给定了 usegraphiccodepage，那么假定导入到图形或双字节字符大对象（DBCLOB）数据字段的数据采用的是图形代码页。而假定其他数据采用的是字符代码页。图形代码页与字符代码页是相关联的。如果指定了 codepage 修饰符，那么 IMPORT 实用程序通过该修饰符来确定字符代码页；如果未指定该修饰符，那么 IMPORT 实用程序通过应用程序的代码页来确定字符代码页。 仅当要恢复的表具有图形数据时，才应将此修饰符与由删除表恢复生成的定界数据文件一起使用。 **限制** 不能对 EXPORT 实用程序创建的 DEL 文件指定 usegraphiccodepage 修饰符，这是因为这些文件中包含只使用一种代码页编码的数据。文件中的双字节字符大对象（DBCLOB）也将忽略 usegraphiccodepage 修饰符。
xmlchar	指定采用字符代码页来 XML 文档进行编码。 处理采用指定的字符代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与字符代码页相匹配，否则将拒绝包含该文档的行。注意，字符代码页就是由文件类型修饰符 codepage 指定的值；如果未指定该修饰符，那么字符代码页就是应用程序代码页。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。
xmlgraphic	指定采用指定的图形代码页来对 XML 文档进行编码。 处理采用特定的图形代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与图形代码页相匹配，否则将拒绝包含该文档的行。注意，图形代码页就是由文件类型修饰符 codepage 指定的值的图形组件；如果未指定该修饰符，那么图形代码页就是应用程序代码页的图形组件。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。 **注**: 如果对 IMPORT 命令指定了 xmlgraphic 修饰符，那么必须采用 UTF-16 代码页来对要导入的 XML 文档进行编码。否则，XML 文档可能会因产生解析错误而被拒绝，或者在将它导入到表中数据被毁坏。

表 16. IMPORT 实用程序的有效文件类型修饰符: ASC（非定界 ASCII）文件格式

修饰符	描述
nochecklengths	如果指定了 nochecklengths，那么即使源数据的列定义超过了目标表列大小，也会尝试导入每一行。如果代码页转换导致源数据缩小，那么可以成功地导入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。
nullindchar=*x*	*x* 是单个字符。将表示空值的字符更改为 *x*。*x* 的缺省值为 Y[3]。 对于 EBCDIC 数据文件，除非字符是英文字母，否则此修饰符将区分大小写。例如，如果将空指示符指定为字母 N，那么 n 也会被识别为空指示符。

表 16. IMPORT 实用程序的有效文件类型修饰符: ASC（非定界 ASCII）文件格式 (续)

修饰符	描述
reclen=*x*	*x* 是一个整数，最大值为 32 767。会读取每行的 *x* 个字符，但未使用换行符来指示行的末尾。

striptblanks	当将数据装入到一个变长字段时，将截断任何尾部空格。如果未指定此选项，那么将保留空格。
	
	在以下示例中，`striptblanks` 将导致 IMPORT 实用程序截断尾部空格:
	`db2 import from myfile.asc of asc`
	`modified by striptblanks`
	`method l (1 10, 12 15) messages msgs.txt`
	`insert into staff`
	
	不能将此选项与 `striptnulls` 同时指定。它们是互斥选项。此选项将替换过时的 `t` 选项，支持此过时选项只是为了保持与早前版本的兼容性。

striptnulls	当将数据装入到一个变长字段时，将截断任何尾部 NULL（0x00 字符）。如果未指定此选项，那么将保留 NULL。 不能将此选项与 `striptblanks` 同时指定。它们是互斥选项。此选项将替换过时的 `padwithzero` 选项，支持此过时选项只是为了保持与早前版本的兼容性。

表 17. IMPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式

修饰符	描述

chardel*x*	*x* 是单个字符串定界符。缺省值是双引号（"）。使用指定的字符而不是使用双引号将字符串引起来[34]。如果您想显式地指定双引号作为字符串定界符，那么应按如下所示指定双引号:
	`modified by chardel""`
	
	也可以指定单引号（'）作为字符串定界符。在以下示例中，`chardel''` 会导致 IMPORT 实用程序将它遇到的任何单引号（'）解释为字符串定界符:
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x* 是一个单字符列定界符。缺省值是逗号（,）。使用指定字符而不是逗号来表示列的末尾[34]。
	
	在以下示例中，`coldel;` 会导致 IMPORT 实用程序将它遇到的任何分号（;）解释为列定界符:
	`db2 import from myfile.del of del`
	`modified by coldel;`
	`messages msgs.txt insert into staff`

decplusblank	加号字符。导致在正的十进制值前面加上空格而不是加号（+）。缺省操作是在正的十进制值前面加上加号。

decpt*x*	*x* 是单个字符，它取代句点作为小数点字符。缺省值是句点（.）。使用指定字符而不是句点作为小数点字符[34]。
	
	在以下示例中，`decpt;` 会导致 IMPORT 实用程序将它遇到的任何分号（;）解释为小数点:
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`decpt; messages msgs.txt insert into staff"`

表 17. IMPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式 (续)

修饰符	描述

delprioritychar	当前，定界符的缺省优先级为记录定界符、字符定界符和列定界符。此修饰符通过将定界符优先级还原为字符定界符、记录定界符和列定界符，来保护依赖于旧的优先级的现有应用程序。语法:
	`db2 import ... modified by delprioritychar ...`
	例如，用以下 DEL 数据文件作为示例:
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	如果指定了 `delprioritychar` 修饰符，那么此数据文件中将只有两行。第二个 <row delimiter> 将被解释为第二行的第一个数据列，而第一个和第三个 <row delimiter> 被解释为实际的记录定界符。如果未指定此修饰符，那么此数据文件中将有三行，每一行都用 <row delimiter> 定界。

keepblanks	保留类型为 CHAR、VARCHAR、LONG VARCHAR 或 CLOB 的每个字段中的前导空格和尾部空格。如果未指定此选项，那么会除去字符定界符外部的所有前导空格和尾部空格，并且会在表中插入 NULL 表示所有空白字段。
nochardel	IMPORT 实用程序将假定在列定界符之间找到的所有字节都是列数据的一部分。字符定界符将被解析为列数据的一部分。如果数据是使用 DB2 导出的，那么不应指定此选项（除非在导出时指定了 nochardel）。提供此修饰符的目的是支持不具有字符定界符的供应商数据文件。未正确使用此修饰符可能会导致数据丢失或毁坏。 不能将此选项与 `chardelx`、`delprioritychar` 或 `nodoubledel` 同时指定。它们是互斥选项。
nodoubledel	不识别双字符定界符。

表 18. IMPORT 实用程序的有效文件类型修饰符: IXF 文件格式

修饰符	描述
forcein	指示实用程序接受数据（即使代码页不匹配也接受），并且阻止代码页之间进行转换。 将检查定长目标字段，以验证它们对于数据来说是否足够大。如果指定了 `nochecklengths`，那么不会进行检查，并且将尝试导入每一行。
indexixf	指示实用程序删除当前已对现有表定义的所有索引，并根据 PC/IXF 文件中的索引定义来创建新的索引。仅当替换表的内容时才使用此选项。不能对视图使用此选项，当指定了 *insert-column* 时也不能指定此选项。
indexschema=*schema*	在创建索引期间，对索引名称使用指定的*模式*。如果未指定*模式*（但是指定了关键字 `indexschema`），那么使用连接用户标识。如果未指定此关键字，那么使用 IXF 文件中的模式。
nochecklengths	如果指定了 `nochecklengths`，那么即使源数据的列定义超过了目标表列大小，也会尝试导入每一行。如果代码页转换导致源数据缩小，那么可以成功地导入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。
forcecreate	指定在导入操作期间返回 SQL3311N 后也应创建表，但是可能会丢失信息或者信息有限。

表 19. 同时使用 codepage 和 usegraphiccodepage 时的 IMPORT 行为

codepage=N	usegraphiccodepage	IMPORT 行为
缺少	缺少	假定文件中的所有数据都采用应用程序代码页。
存在	缺少	假定文件中的所有数据都采用代码页 N。 **警告:** 如果 N 是单字节代码页，那么将图形数据导入到数据库中时将毁坏该数据。
缺少	存在	假定文件中的字符数据都采用应用程序代码页。假定图形数据要采用应用程序图形数据的代码页。 如果应用程序代码页是单字节，那么假定所有数据都采用应用程序代码页。 **警告:** 如果应用程序代码页是单字节，那么图形数据在导入到数据库中将毁坏数据，即使数据库中包含图形列亦如此。
存在	存在	假定字符数据采用代码页 N。假定图形数据采用图形代码页 N。 如果 N 是单字节或双字节代码页，那么假定所有数据都采用代码页 N。 **警告:** 如果 N 是单字节代码页，那么将图形数据导入到数据库中时将毁坏该数据。

注:

1. 如果您尝试将不受支持的文件类型与 MODIFIED BY 选项配合使用，IMPORT 实用程序不会发出警告。如果尝试这样做，导入操作将失败，并且会返回错误代码。
2. 日期格式字符串两边必须具有双引号。字段分隔符不能包含下列任何字符: a-z、A-Z 和 0-9。字段分隔符不应与 DEL 文件格式中的字符定界符或字段定界符相同。如果元素的开始和结束位置是明确的，那么字段分隔符是可选的。如果使用了诸如 D、H、M 或 S 之类的元素（取决于修饰符），那么由于条目长度是可变的，因此可能存在不明确性。

   对于时间戳记格式，必须要注意避免月份描述符与分钟描述符之间的不明确性，这是因为它们都使用字母 M。月份字段必须与其他日期字段相邻。而分钟字段必须与其他时间字段相邻。以下是一些不明确的时间戳记格式:

   ```
   "M"（既可能是月份，也可能是分钟）
      "M:M"（无法区分哪个是月份，哪个是分钟）
      "M:YYYY:M"（两者都将被解释为月份。）
      "S:M:YYYY"（与时间值和日期值都相邻）
   ```

   在不明确的情况下，实用程序将报告一条错误消息，并且操作将失败。

   以下是一些明确的时间戳记格式:

   ```
   "M:YYYY"（表示月份）
      "S:M"（表示分钟）
      "M:YYYY:S:M"（前者表示月份，后者表示分钟）
      "M:H:YYYY:M:D"（前者表示分钟，后者表示月份）
   ```

   在某些字符（例如，双引号和反斜杠）前面必须添加转义字符（例如，@2329。

3. 为 chardel、coldel 或 decpt 文件类型修饰符指定的字符值必须采用源数据的代码页来指定。

   可以使用 xJJ 或 0xJJ 语法来指定字符代码点（而不是字符符号）。其中 JJ 是代码点的十六进制表示法。例如，要指定 # 字符作为列定界符，可使用下列方法之一:

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

4. *移动数据时的定界符注意事项*列示了可以用作定界符的字符存在的限制。
5. 在导入到昵称时，将不允许使用下列文件类型修饰符:
   - `indexixf`
   - `indexschema`
   - `dldelfiletype`
   - `nodefaults`
   - `usedefaults`
   - `no_type_idfiletype`
   - `generatedignore`
   - `generatedmissing`
   - `identityignore`
   - `identitymissing`
   - `lobsinfile`
6. XML 列不支持 **WSF** 文件格式。
7. XML 列不支持 CREATE 方式。
8. 所有 XML 数据必须位于与主数据文件分隔开的 XML 文件中。主数据文件中的每个 XML 列都必须存在 XML 数据说明符（XDS）（或者空值）。
9. 除非指定了 XMLCHAR 或 XMLGRAPHIC 文件类型修饰符，否则将假定 XML 文档采用 Unicode 格式或者包含一个包括编码属性的声明标记。
10. 将拒绝包含结构不当的文档的行。
11. 如果指定了 XMLVALIDATE 选项，那么在插入成功验证了其匹配模式的文档时，会使用模式信息对这些文档进行注释。如果行中包含的文档针对其匹配模式验证失败，那么将拒绝这些行。要成功执行验证，调用 IMPORT 实用程序的用户拥有的特权必须至少包括下列其中一项权限或特权:
    - SYSADM 或 DBADM 权限
    - 对验证时要使用的 XML 模式的 USAGE 特权
12. 导入到包含隐式隐藏的 ROW CHANGE TIMESTAMP 列的表中时，将不采用该列的隐式隐藏属性。因此，如果列的数据在要导入的数据中不存在，且不存在任何显式列列表，那么*必须*在导入命令中指定 rowchangetimestampmissing 文件类型修饰符。

使用 ADMIN_CMD 过程的 IMPORT 命令

将具有受支持文件格式的外部文件中的数据插入表、层次结构、视图或昵称中。另一种速度更快的方法是使用 LOAD 实用程序，但是 LOAD 实用程序不支持在层次结构级别装入数据。

指向 第 92 页的『IMPORT 实用程序的文件类型修饰符』的快速链接。

权限

- 使用 INSERT 选项进行 IMPORT 时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对参与的每个表、视图或昵称的 CONTROL 特权
  - 对参与的每个表或视图的 INSERT 和 SELECT 特权
- 使用 INSERT_UPDATE 选项 IMPORT（导入）到现有表时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对参与的每个表、视图或昵称的 CONTROL 特权
  - 对参与的每个表或视图的 INSERT、SELECT、UPDATE 和 DELETE 特权
- 使用 REPLACE 或 REPLACE_CREATE 选项 IMPORT（导入）到现有表时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对表或视图的 CONTROL 特权
  - 对表或视图的 INSERT、SELECT 和 DELETE 特权
- 使用 CREATE 或 REPLACE_CREATE 选项 IMPORT（导入）到新表时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对数据库的 CREATETAB 权限、对表空间的 USE 特权以及下列其中一项权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果该表的隐式或显式模式名称不存在的话）
    - 对模式的 CREATEIN 特权（如果该表的模式名称引用现有模式的话）
- 使用 CREATE 或 REPLACE_CREATE 选项 IMPORT（导入）到不存在的层次结构时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*
  - 对数据库的 CREATETAB 权限、对表空间的 USE 特权以及下列其中一项权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果表的模式名称不存在的话）
    - 对模式的 CREATEIN 特权（如果表的模式存在的话）
    - 对层次结构中的每个子表的 CONTROL 特权（如果使用了整个层次结构上的 REPLACE_CREATE 选项的话）
- 使用 REPLACE 选项 IMPORT（导入）到现有层次结构时需要具有下列权限之一:
  - *sysadm*
  - *dbadm*

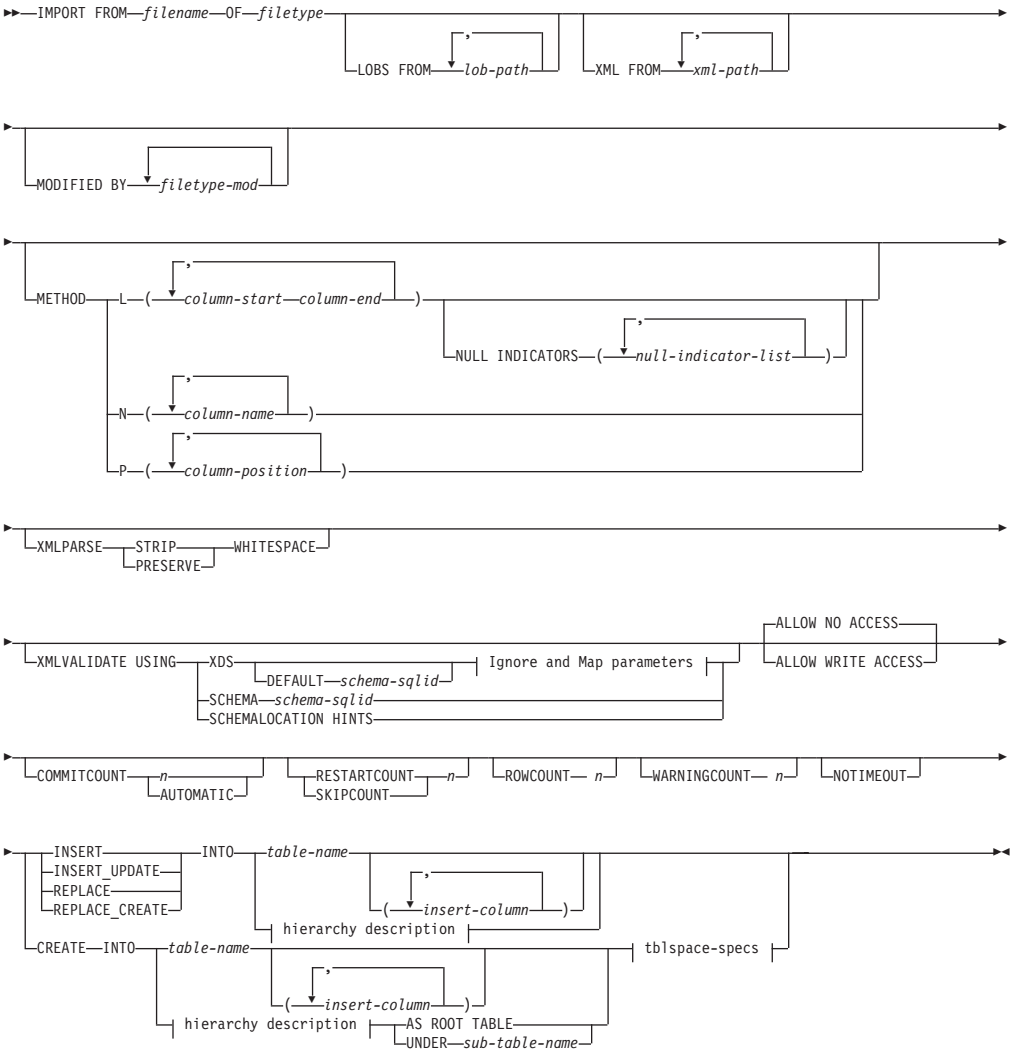
– 对层次结构中每个子表的 CONTROL 特权

- 要将数据导入到包含受保护列的表中，会话授权标识必须拥有允许对该表中所有受保护列执行写访问的 LBAC 凭证。否则，导入将失败并且返回错误（SQLSTATE 42512）。
- 要将数据导入到包含受保护行的表中，会话授权标识必须拥有满足下列条件的 LBAC 凭证:
  – 它是用于保护表的安全策略的一部分
  – 已将它授予会话授权标识以进行写访问

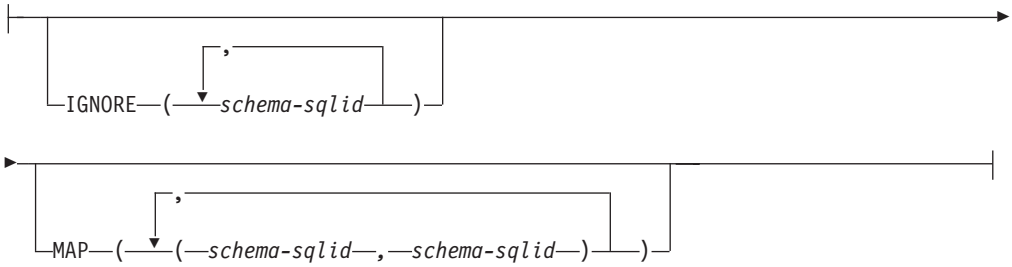
  要插入的行上的标号、用户的 LBAC 凭证、安全策略定义和 LBAC 规则共同确定行上的标号。
- 如果指定了 REPLACE 或 REPLACE_CREATE 选项，那么会话授权标识必须有权删除该表。

必需的连接

命令语法

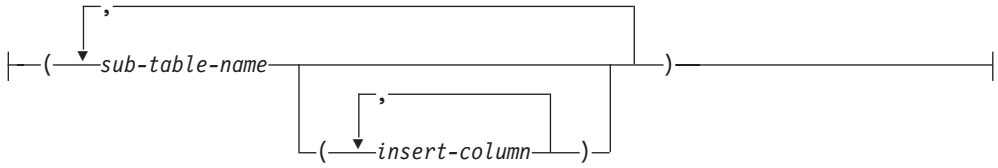


Ignore and Map parameters:



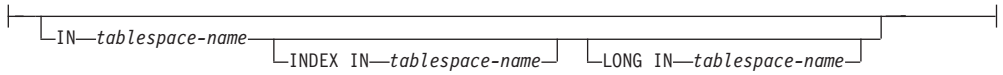
hierarchy description:

sub-table-list:



traversal-order-list:

tblspace-specs:



命令参数

ALL TABLES

这是仅用于层次结构的隐式关键字。当导入层次结构时，缺省情况是导入按遍历顺序指定的所有表。

ALLOW NO ACCESS

以脱机方式运行导入操作。在插入任何行前，就会获得对目标表的互斥（X）锁定。这将阻止并发应用程序访问表数据。这是缺省导入行为。

ALLOW WRITE ACCESS

以联机方式运行导入操作。插入第一行时，将获得对目标表的意向互斥（IX）锁定。这允许并发阅读器和写程序访问表数据。联机方式与 REPLACE、CREATE 或 REPLACE_CREATE 导入选项不兼容。不支持将联机

方式与缓冲插入一起使用。导入操作将定期落实已插入的数据，以防止锁定升级为表锁定，并且避免用尽活动日志空间。即使未使用 COMMITCOUNT 选项，也将执行这些落实操作。在每次落实时，导入操作都将失去它的 IX 表锁定，但是在落实后又将尝试重新获取锁定。当导入到昵称时，此参数是必需的，并且必须为 COMMITCOUNT 指定一个有效数（AUTOMATIC 不被视为有效选项）。

AS ROOT TABLE

创建一个或多个子表作为独立表层次结构。

COMMITCOUNT *n* **| AUTOMATIC**

导入 *n* 条记录后，将执行 COMMIT。当指定了数目 *n* 时，IMPORT 实用程序在每导入 *n* 条记录后就会执行 COMMIT。使用复合插入时，会将用户指定的落实频率 *n* 向上舍入为复合计数值的第一个整数倍数。当指定了 AUTOMATIC 时，IMPORT 实用程序将在内部确定需要执行落实的时间。该实用程序将因下列任一原因而进行落实:

- 为了避免用尽活动日志空间
- 为了避免使锁定从行级别升级到表级别

如果指定了 ALLOW WRITE ACCESS 选项，而未指定 COMMITCOUNT 选项，那么 IMPORT 实用程序将执行落实操作，如同指定了 COMMITCOUNT AUTOMATIC 一样。

如果在插入或更新记录时 IMPORT 命令遇到了 SQL0964C（事务日志已满）并且指定了 COMMITCOUNT *n*，那么 IMPORT 将尝试通过执行无条件的落实，然后重新尝试插入或更新记录来解决此问题。如果这样做不能帮助您解决日志已满的情况（也就是因对数据库执行其他活动而使日志已满的情况），那么 IMPORT 命令将像预期那样失败，但是落实的行数可能不是 COMMITCOUNT *n* 值的倍数。可以使用 RESTARTCOUNT 或 SKIPCOUNT选项来避免处理已经落实的那些行。

CREATE

注: 不推荐使用 CREATE 参数，将来的发行版中可能会将其除去。有关其他详细信息，请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”。

采用数据库的代码页来创建表定义和行内容。如果数据是从 DB2 表、子表或层次结构中导出的，那么将创建索引。如果将此选项应用于层次结构，并且数据是从 DB2 中导出的，那么还将创建类型层次结构。此选项只能用于 IXF 文件。

当导入到昵称时，此参数无效。

注: 如果数据是从 MVS 主机数据库中导出的，并且它包含一些 LONGVAR 字段，而按照页大小计算的这些字段的长度超过了 254，那么 CREATE 操作可能会因行太长而失败。请参阅『重新创建导入的表』，以获取限制列表。在这种情况下，应该手动创建表，并且应该调用指定了 INSERT 的 IMPORT，或者应该使用 LOAD 命令。

DEFAULT *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。通过 DEFAULT 子句指

定的模式标识: 当已导入的 XML 文档的 XML 数据说明符（XDS）不包含用于标识 XML 模式的 SCH 属性时要用于验证的模式。

DEFAULT 子句优先于 IGNORE 和 MAP 子句。如果 XDS 满足 DEFAULT 子句，那么将忽略 IGNORE 和 MAP 规范。

FROM *filename*

HIERARCHY

指定要导入分层数据。

IGNORE *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。如果 SCH 属性标识了一种或多种模式，那么 IGNORE 子句指定这些模式中要忽略的模式列表。如果 SCH 属性存在于已导入的 XML 文档的 XML 数据说明符中，并且由 SCH 属性标识的模式包含在要忽略的模式列表中，那么将不会对已导入的 XML 文档进行模式验证。

如果某一模式是在 IGNORE 子句中指定的，那么该模式不能存在于 MAP 子句中模式对的左边。

IGNORE 子句仅适用于 XDS。如果 IGNORE 子句指定了由 MAP 子句映射的模式，那么以后不会忽略该模式。

IN *tablespace-name*

确定将在其中创建表的表空间。该表空间必须存在，并且必须是 REGULAR（常规）表空间。如果未指定其他表空间，那么所有表部件都将存储在此表空间中。如果未指定此子句，那么将在由授权标识创建的表空间中创建表。如果找不到表空间，那么会将该表放入缺省表空间 USERSPACE1 中。如果已经删除了 USERSPACE1，那么表创建操作将失败。

INDEX IN *tablespace-name*

确定将在其中对表创建索引的表空间。仅当在 IN 子句中指定的主表空间是一个 DMS 表空间时，才允许使用此选项。指定的表空间必须存在，并且必须是 REGULAR 或 LARGE DMS 表空间。

注: 仅当创建了表时，才能指定哪个表空间将包含索引。

insert-column

指定要将数据插入到的表或视图中的列名。

INSERT

将已导入的数据添加至表，但不更改现有表数据。

INSERT_UPDATE

将已导入的数据行添加至目标表，或者使用相匹配的主键来更新（目标表的）现有行。

INTO *table-name*

指定要将数据导入到的数据库表。此表不能是系统表、已声明临时表或者总结表。

应该使用标准表名或者非限定表名时，除了使用早期版本的服务器的情况以外，可以对 INSERT、INSERT_UPDATE 或 REPLACE 使用别名。标准表名的格式为: *schema.tablename*。*schema* 是创建表时所使用的用户名。

LOBS FROM *lob-path*

LOB 数据文件的名称存储在将装入到 LOB 列的那列的主数据文件（ASC、DEL 或 IXF）中。最多可指定 999 个路径。这将隐式激活 LOBSINFILE 行为。

当导入到昵称时，此参数无效。

LONG IN *tablespace-name*

确定将用来存储任何长列（LONG VARCHAR、LONG VARGRAPHIC、LOB 数据类型或者将任何这些数据类型作为源类型的单值类型）的值的表空间。仅当在 IN 子句中指定的主表空间是一个 DMS 表空间时，才允许使用此选项。该表空间必须存在，并且必须是 LARGE DMS 表空间。

MAP *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。使用 MAP 子句来指定要使用的替代模式，这些模式将取代由已导入的每个 XML 文档的 XML数据说明符（XDS）的 SCH 属性所指定的那些模式。MAP 子句指定一个或多个模式对的列表，而每个模式对都表示一个模式与另一个模式之间的映射。模式对中的第一个模式表示 XDS 中的 SCH 属性引用的模式。模式对中的第二个模式表示应该用来执行模式验证的模式。

如果某一模式存在于 MAP 子句中的模式对的左边，那么不能在 IGNORE 子句中也指定该模式。

一旦应用了模式对映射，其结果就是最终结果。映射操作不是过渡操作，因此，以后不会将所选择的模式应用于另一个模式对映射。

不能多次映射同一模式，这就意味着该模式不会多次出现在一个模式对的左边。

METHOD

L 指定要将数据导入到的列的起始列号和结束列号。列号就是与一个数据行的开头的字节偏移量。它是从 1 开始计数的。

注: 此方法只能用于 ASC 文件，并且对于该文件类型是唯一有效的选项。

N 指定要导入的数据文件中的列名。这些列名的大小写必须与系统目录中相应名称的大小写相匹配。不可空的每个表列在 METHOD N 列表中都应该具有相应的条目。例如，给定数据字段 F1、F2、F3、F4、F5 和 F6，以及表列 C1 INT、C2 INT NOT NULL、C3 INT NOT NULL 和 C4 INT，那么 method N (F2, F1, F4, F3) 是有效请求，而 method N (F2, F1) 是无效请求。

注: 此方法只能用于 IXF 文件。

P 指定要导入的输入数据字段的字段编号。

注: 此方法只能用于 IXF 或 DEL 文件，并且对于 DEL 文件类型是唯一有效的选项。

MODIFIED BY *filetype-mod*

指定文件类型修饰符选项。请参阅 第 92 页的『IMPORT 实用程序的文件类型修饰符』。

NOTIMEOUT

指定 IMPORT 实用程序在等待锁定时不会超时。此选项取代了数据库配置参数 *locktimeout*。其他应用程序将不受影响。

NULL INDICATORS *null-indicator-list*

仅当指定了 METHOD L 参数时，才能使用此选项。即，输入文件是 ASC 文件。空指示符列表是用于指定每个空指示符字段的正整数的列表，各个正整数之间用逗号隔开。列号是空指示符字段与一个数据行的开头之间的字节偏移量。对于 METHOD L 参数中定义的每个数据字段，在空指示符列表中必须有一个其对应的条目。如果列号为 0，那么表示相应的数据字段中始终包含数据。

如果空指示符列中的值为 `Y`，那么表示列数据为 NULL。如果空指示符列中的值是除了 `Y` 之外的任何字符，那么表示列数据不是 NULL，并且将导入由 METHOD L 选项指定的列数据。

可通过将 MODIFIED BY 选项与文件类型修饰符 `nullindchar` 一起使用来更改空指示符。

OF *filetype*

指定输入文件中数据的格式:

- ASC（非定界 ASCII 格式）
- DEL（定界 ASCII 格式），多种数据库管理器和文件管理器程序使用此格式
- WSF（工作表格式），下列程序使用此格式:
  - Lotus 1-2-3
  - Lotus Symphony
- IXF（PC 版本的集成交换格式）是一种专门供 DB2 使用的二进制格式。

当导入到昵称时，WSF 文件类型不受支持。

REPLACE

通过截断数据对象来删除表中现有的所有数据，然后插入已导入的数据。表定义和索引定义不会发生更改。仅当表存在时才能使用此选项。如果在层次结构之间移动数据时使用了此选项，那么只能替换整个层次结构的数据，而不能替换单个子表的数据。

当导入到昵称时，此参数无效。

此选项不支持 CREATE TABLE 语句的 NOT LOGGED INITIALLY (NLI) 子句或者 ALTER TABLE 语句的 ACTIVE NOT LOGGED INITIALLY 子句。

如果包含 REPLACE 选项的导入操作与调用了 NLI 子句的 CREATE TABLE 或 ALTER TABLE 语句在同一事务内执行，那么此导入操作不能识别该 NLI 子句。将记录所有插入操作。

变通方法 1

使用 DELETE 语句删除表的内容，然后使用 INSERT 语句调用导入操作

变通方法 2

删除表后再重新创建表，然后使用 INSERT 语句调用导入操作。

DB2 通用数据库版本 7 和 DB2 UDB 版本 8 存在此局限性

REPLACE_CREATE

注: 不推荐使用 REPLACE_CREATE 参数，将来的发行版中可能会将其除去。有关其他详细信息，请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”。

如果表存在，那么通过截断数据对象来删除表中现有的所有数据，然后插入已导入的数据，但不更改表定义或索引定义。

如果表不存在，那么采用数据库的代码页创建表和索引定义以及行内容。请参阅*重新创建导入的表*，以获取限制列表。

此选项只能用于 IXF 文件。如果在层次结构之间移动数据时使用了此选项，那么只能替换整个层次结构的数据，而不能替换单个子表的数据。

当导入到昵称时，此参数无效。

RESTARTCOUNT *n*

指定要从第 *n* + 1 条记录开始执行导入操作。将跳过前 *n* 条记录。此选项的功能相当于 SKIPCOUNT。RESTARTCOUNT 和 SKIPCOUNT 是互斥的。

ROWCOUNT *n*

指定要导入（插入或更新）文件中的 *n* 条物理记录。允许用户从文件中仅导入 *n* 行（从由 SKIPCOUNT 或 RESTARTCOUNT 选项确定的记录开始算起）。如果未指定 SKIPCOUNT 或 RESTARTCOUNT 选项，那么会导入前 *n* 行。如果指定了 SKIPCOUNT *m* 或 RESTARTCOUNT *m*，那么将导入从第 *m*+1 行到第 *m*+*n* 行。使用复合插入时，会将用户指定的 ROWCOUNT *n* 向上舍入为复合计数值的第一个整数倍数。

SKIPCOUNT *n*

指定要从第 *n* + 1 条记录开始执行导入操作。将跳过前 *n* 条记录。此选项的功能相当于 RESTARTCOUNT。SKIPCOUNT 和 RESTARTCOUNT 是互斥的。

STARTING *sub-table-name*

这是仅用于层次结构的一个关键字，请求缺省顺序，从 *sub-table-name* 开始。对于 PC/IXF 文件，缺省顺序是存储在输入文件中的顺序。对于 PC/IXF 文件格式，缺省顺序是唯一有效的顺序。

sub-table-list

对于使用 INSERT 或 INSERT_UPDATE 选项的类型表，使用子表名的列表来指示要将数据导入到的子表。

traversal-order-list

对于使用 INSERT、INSERT_UPDATE 或 REPLACE 选项的类型表，使用子表名的列表来指示导入层次结构中的子表的遍历顺序。

UNDER *sub-table-name*

指定用于创建一个或多个子表的父表。

WARNINGCOUNT *n*

在发出 *n* 个警告后停止导入操作。如果希望不产生警告，那么设置此参数，但是需要验证使用的是正确的文件和表。如果指定了不正确的导入文件或目标表，那么 IMPORT 实用程序将对它试图导入的每一行生成一个警告，这将导致导入失败。如果 *n* 为 0，或者未指定此选项，那么无论发出多少个警告，都将继续执行导入操作。

XML FROM *xml-path*

指定一个或多个包含 XML 文件的路径。

XMLPARSE

指定如何解析 XML 文档。如果未指定此选项，那么将由 CURRENT XMLPARSE OPTION 专用寄存器的值来确定 XML 文档的解析行为。

STRIP WHITESPACE

指定在解析 XML 文档时要去掉空格。

PRESERVE WHITESPACE

指定在解析 XML 文档时不去掉空格。

XMLVALIDATE

指定在适当的情况下将针对某一模式来验证 XML 文档。

USING XDS

将针对由主数据文件中的 XML 数据说明符（XDS）标识的 XML 模式来验证 XML 文档。缺省情况下，如果使用 USING XDS 子句调用了 XMLVALIDATE 选项，那么将由 XDS 的 SCH 属性来确定用来执行验证的模式。如果 XDS 中不存在 SCH 属性，那么除非缺省模式是由 DEFAULT 子句指定的，否则将不会进行模式验证。

可以使用 DEFAULT、IGNORE 和 MAP 子句来修改模式确定行为。这三个可选子句直接应用于 XDS 的指定，但是它们不会互相应用。例如，如果由于 DEFAULT 子句指定了某一模式而选择了该模式，那么，即使 IGNORE 子句也指定了该模式，该模式仍不会被忽略。类似，如果由于将某一模式指定为 MAP 子句对的第一部分而选择了该模式，那么，即使在另一个 MAP 子句对的第二部分中也指定了该模式，该模式仍不会被重新映射。

USING SCHEMA *schema-sqlid*

将针对具有指定的 SQL 标识的 XML 模式来验证 XML 文档。在这种情况下，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

USING SCHEMALOCATION HINTS

将针对由源 XML 文档中的 XML 模式位置提示标识的模式来验证 XML 文档。如果在 XML 文档中找不到 schemaLocation 属性，那么将不执行验证。指定 USING SCHEMALOCATION HINTS 子句时，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

请参阅下面的 XMLVALIDATE 选项示例。

使用说明

在开始执行导入操作前，务必完成所有表操作并释放所有锁定。这可以通过在关闭使用 WITH HOLD 打开的所有游标前发出 COMMIT，或者通过发出 ROLLBACK 来完成。

IMPORT 实用程序使用 SQL INSERT 语句将一些行添加至目标表。该实用程序将对输入文件中的每行数据发出一个 INSERT 语句。如果 INSERT 语句失败，那么将导致执行以下两个操作之一:

- 如果后续 INSERT 语句可能会成功，那么会将警告消息写入消息文件中，但是将继续进行处理。

- 如果后续 INSERT 语句可能会失败，并且数据库可能会被破坏，那么会将错误消息写入消息文件并停止处理。

在执行 REPLACE 或 REPLACE_CREATE 操作期间删除了旧的行后，该实用程序将执行自动落实（COMMIT）。因此，如果系统失败，或者在截断表对象后应用程序中断了数据库管理器，那么所有旧数据都会丢失。确保在使用这些选项前不再需要旧数据。

如果在执行 CREATE、REPLACE 或 REPLACE_CREATE 操作期间日志已满，那么该实用程序将对已插入的记录执行自动落实（COMMIT）。如果系统失败，或者在自动落实（COMMIT）后应用程序中断了数据库管理器，那么具有初始数据的表将保留在数据库中。使用 REPLACE 或 REPLACE_CREATE 选项来重新运行整个导入操作，或者在 RESTARTCOUNT 参数设置为已成功导入的行数的情况下执行 INSERT 操作。

缺省情况下，不会对 INSERT 或 INSERT_UPDATE 选项执行自动落实（COMMIT）。但是，如果 COMMITCOUNT 参数不为零，那么将执行自动落实（COMMIT）。如果未执行自动落实（COMMIT），当日志已满时就会导致回滚（ROLLBACK）。

如果存在下列任何一种情况，脱机导入将不执行自动落实（COMMIT）：

- 目标是视图而不是表
- 使用了复合插入
- 使用了缓冲插入

缺省情况下,联机导入时将执行自动落实（COMMIT），以同时释放活动日志空间和锁定列表。仅当将 COMMITCOUNT 值指定为 0 时，才不会执行自动落实（COMMIT）。

每当 IMPORT 实用程序执行落实（COMMIT）时，会将两条消息写入消息文件中：一条消息指示要落实的记录数；在成功执行落实（COMMIT）后将写入另一条消息。当在失败后重新启动导入操作时，请指定要跳过的记录数，此数目是在上一次成功执行落实（COMMIT）时确定的。

IMPORT 实用程序将接受带有次要的不兼容问题的输入数据（例如，可通过使用填充或截断来导入的字符数据，以及可以使用另一种数字数据类型来导入的数字数据），但是不会接受具有主要的不兼容问题的数据。

如果一个对象表具有除它本身之外的任何从属项，或者如果对象视图的基本表具有任何从属项（包括它本身），那么不能对该对象表或对象视图执行 REPLACE 或 REPLACE_CREATE。要替换这样的一个表或视图，请执行以下操作:

1. 删除该表是父表的所有外键。
2. 运行 IMPORT 实用程序。
3. 改变该表以重新创建外键。

如果在重新创建外键时发生了错误，那么修改数据以保持引用完整性。

根据 PC/IXF 文件重新创建表时，将不保留引用约束和外键定义。如果数据先前是使用 `SELECT *` 导出的，那么会保留主键定义。

在导入到远程数据库时，要求服务器上有足够的磁盘空间来复制输入数据文件、输出消息文件以及满足数据库可能增大的需求。

如果对远程数据库运行导入操作，并且输出消息文件很长（超过 60 KB），那么返回给客户机上的用户的消息文件可能会在导入操作过程中丢失消息。但始终会保留前 30 KB 的消息信息和最后 30 KB 的消息信息。

如果 PC/IXF 文件位于硬盘驱动器而不是软盘上，那么将 PC/IXF 文件导入到远程数据库中时速度会更快。

必须存在数据库表或层次结构，才能导入采用 **ASC**、**DEL** 或 **WSF** 文件格式的数据；但是，如果表尚不存在，那么 IMPORT CREATE 或 IMPORT REPLACE_CREATE 在从 PC/IXF 文件中导入数据时将创建表。对于类型表，IMPORT CREATE 可以创建类型层次结构以及表层次结构。

应该使用 PC/IXF 导入来在数据库之间移动数据（包括分层数据）。如果包含行分隔符的字符数据已导出到定界 ASCII（DEL）文件并且已由文本传输程序处理，那么包含行分隔符的字段将收缩或展开。如果从同一台客户机可以访问源数据库和目标数据库，那么不需要执行文件复制步骤。

默认 ASC 和 DEL 文件中的数据采用的是执行导入操作的客户机应用程序的代码页。当导入采用其他代码页的数据时，建议使用 PC/IXF 文件（这些文件允许使用其他代码页）。如果 PC/IXF 文件与 IMPORT 实用程序采用相同的代码页，那么像常规应用程序一样进行处理。如果它们使用不同的代码页，并且指定了 FORCEIN 选项，那么 IMPORT 实用程序将认为 PC/IXF 文件中的数据与执行导入操作的应用程序具有相同的代码页。即使这两种代码页之间存在转换表，也会发生这种情况。如果它们使用不同的代码页，并且未指定 FORCEIN 选项，代码页之间还存在转换表，那么 PC/IXF 文件中的所有数据都将从文件代码页转换为应用程序代码页。如果它们使用不同的代码页，并且未指定 FORCEIN 选项，代码页之间不存在转换表，那么导入操作将失败。这仅适用于采用 AIX 操作系统的 DB2 客户机上的 PC/IXF 文件。

对于 8 KB 页（接近 1012 列这一限制）上的表对象，导入 PC/IXF 数据文件可能会导致 DB2 返回错误，这是因为已经超过了 SQL 语句的最大大小。仅当列类型为 CHAR、VARCHAR 或 CLOB 时才会发生这种情况。该限制并不适用于导入 **DEL** 或 **ASC** 文件这种情况。如果使用 PC/IXF 文件来创建新表，那么替代方法是使用 db2look 来转储用于创建该表的 DDL 语句，然后通过 CLP 发出该语句。

可以使用 DB2 Connect 将数据导入到 DRDA 服务器（例如，DB2 OS/390 版、DB2 VM 和 VSE 版以及 DB2 OS/400 版）中。仅支持 PC/IXF 导入（INSERT 选项）。还支持 RESTARTCOUNT 参数，但不支持 COMMITCOUNT 参数。

对类型表使用 CREATE 选项时，将创建在 PC/IXF 文件中定义的每个子表；不能改变子表定义。对类型表使用除 CREATE 之外的其他选项时，遍历顺序列表允许用户指定遍历顺序；因此，遍历顺序列表必须与在执行导出操作期间所使用的遍历顺序列表相匹配。对于 PC/IXF 文件格式，用户只需要指定目标子表名，并使用文件中存储的遍历顺序。

可以使用 IMPORT 实用程序来恢复先前已导出到 PC/IXF 文件的表。该表将返回到导出它时所处的状态。

不能将数据导入到系统表、已声明临时表或总结表。

不能通过 import 实用程序创建视图。

支持导入多部件 PC/IXF 文件，该文件的各个部件是从 Windows 系统复制到 AIX 系统的。仅必须在 IMPORT 命令中指定第一个文件的名称。例如，`IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1`。文件 data.002 等应该位于 `data.ixf` 所在的目录中。

在 Windows 操作系统上:

- 不支持导入按逻辑划分的 PC/IXF 文件。
- 不支持导入错误格式的 PC/IXF 或 WSF 文件。

采用内部格式的安全标号可能包含换行符。如果导入使用 DEL 文件格式的文件，那么这些换行符可能会被误认为定界符。如果发生此问题，请通过在 IMPORT 命令中指定文件类型修饰符 `delprioritychar` 来对定界符使用较旧的缺省优先级。

联合注意事项

使用 IMPORT 命令和 INSERT、UPDATE 或 INSERT_UPDATE 命令参数时，必须确保您对参与的昵称具有 CONTROL 特权。必须确保您在执行导入操作时想使用的昵称已存在。还存在一些应注意的限制，在 IMPORT 命令参数部分说明了这些限制。

某些数据源（例如 ODBC）不支持导入到昵称。

IMPORT 实用程序的文件类型修饰符

表 20. IMPORT 实用程序的有效文件类型修饰符: 所有文件格式

修饰符	描述
compound=*x*	*x* 是一个 1 到 100 之间（包括 1 和 100 在内）的数字。使用无原子的复合 SQL 来插入数据，并且每次都将尝试使用 *x* 语句。 如果指定了此修饰符，而事务日志不是足够大，那么导入操作将失败。事务日志必须足够大，用来容纳由 COMMITCOUNT 指定的行数；如果未指定 COMMITCOUNT，那么用来容纳数据文件中的行数。因此，建议指定 COMMITCOUNT 选项以避免事务日志溢出。 此修饰符与 INSERT_UPDATE 方式、分层表和下列修饰符不兼容: `usedefaults`、`identitymissing`、`identityignore`、`generatedmissing` 和 `generatedignore`。
generatedignore	此修饰符通知 IMPORT 实用程序: 所有生成列的数据在数据文件中都存在，但是应该忽略这些数据。这将导致所有生成列值都由该实用程序生成。此修饰符不能与 `generatedmissing` 修饰符一起使用。
generatedmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含生成列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。此修饰符不能与 `generatedignore` 修饰符一起使用。
identityignore	此修饰符通知 IMPORT 实用程序: 标识列的数据在数据文件中已存在，但是应该忽略该数据。这将导致所有标识值都由该实用程序生成。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 `identitymissing` 修饰符一起使用。
identitymissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含标识列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。此修饰符不能与 `identityignore` 修饰符一起使用。

表 20. IMPORT 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
lobsinfile	*lob-path* 指定包含 LOB 数据的文件所在的路径。 每个路径都至少包含这样一个文件: 该文件至少包含数据文件中的“Lob 位置说明符”（LLS）所指向的一个 LOB。对于存储在 LOB 文件路径中的文件，LLS 就是对这些文件中的 LOB 所在位置的字符串表示。LLS 的格式为 *filename.ext.nnn.mmm/*，其中 *filename.ext* 是包含 LOB 的文件名，*nnn* 是文件中的 LOB 的偏移量（以字节计），*mmm* 是 LOB 的长度（以字节计）。例如，如果 `db2exp.001.123.456/` 字符串存储在数据文件中，那么 LOB 位于 db2exp.001 文件中偏移量为 123 的位置，其长度为 456 字节。 使用“lobsinfile”修饰符时，LOBS FROM 子句指定 LOB 文件所在的位置。LOBS FROM 子句将隐式激活 LOBSINFILE 行为。导入数据时，LOBS FROM 子句将要从其中搜索 LOB 文件的路径列表传递给 IMPORT 实用程序。 要指示一个空 LOB，应将大小输入为 -1。如果将大小指定为 0，那么会将它视作长度为 0 的 LOB。对于长度为 -1 的 LOBS，将忽略偏移量和文件名。例如，空 LOB 的 LLS 可能是 db2exp.001.7.-1/。
no_type_id	仅当导入到单个子表中时才有效。典型用途是从常规表中导出数据，然后（使用此修饰符）调用导入操作来将该数据转换到单个子表中。
nodefaults	如果未显式地指定目标表列的源列，而表列又不可空，那么不会装入缺省值。在不使用此选项的情况下，如果未显式地指定其中一个目标表列的源列，那么将发生下列情况之一: • 如果可以为一列指定缺省值，那么将装入缺省值 • 如果该列可空，且不能为该列指定缺省值，那么将装入 NULL • 如果该列不可空，但不能为该列指定缺省值，那么将返回错误，并且该实用程序将停止处理。
norowwarnings	抑制关于被拒绝行的所有警告。
rowchangetimestampignore	此修饰符通知 IMPORT 实用程序: ROW CHANGE TIMESTAMP 列的数据在数据文件中已存在，但是应该忽略该数据。这将导致该实用程序生成所有 ROW CHANGE TIMESTAMP。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 `rowchangetimestampmissing` 修饰符一起使用。
rowchangetimestampmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含 ROW CHANGE TIMESTAMP 列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。此修饰符不能与 `rowchangetimestampignore` 修饰符一起使用。
seclabelchar	指示输入源文件中的安全标号采用安全标号值的字符串格式，而不是采用缺省编码数字格式。当装入每个安全标号时，IMPORT 实用程序会将它们转换为内部格式。如果字符串的格式不正确，那么将不装入该行，并且将返回警告（SQLSTATE 01H53）。如果字符串并不表示作为用于保护表的安全策略的一部分的有效安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3243W）。 如果指定了 seclabelname 修饰符，那么不能指定此修饰符，否则导入会失败并且将返回错误（SQLCODE SQL3525N）。

表 20. IMPORT 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
seclabelname	指示输入源文件中的安全标号是由它们的名称指示的，而不是由缺省编码数字格式指示的。如果存在名称，那么 IMPORT 实用程序会将它转换为相应的安全标号。如果不存在具有所指示的（用于保护表的）安全策略名称的安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3244W）。 如果指定了 seclabelchar 修饰符，那么不能指定此修饰符，否则导入会失败并且将返回错误（SQLCODE SQL3525N）。 **注**: 如果文件类型是 ASC，那么安全标号名称后面的任何空格都将被解释为该名称的一部分。为了避免此问题，可以使用文件类型修饰符 striptblanks 来确保除去空格。
usedefaults	如果已经指定了目标表列的源列，但是源列中不包含一个或多个行实例的数据，那么将装入缺省值。缺少的数据的示例如下: • 对于 DEL 文件: 为列值指定了两个相邻的列定界符（“,,”）或者两个相邻的列定界符之间还有任意数目的空格（“, ,”）。 • 对于 DEL/ASC/WSF 文件: 没有足够的列数或者对于原始规范来说不是足够长的行。 **注**: 对于 ASC 文件，并不将列值为 NULL 认为是显式丢失，也不会用缺省值来代替 NULL 列值。NULL 列值是由数字、日期、时间和时间戳记列的所有空格字符来表示的，或者是通过对任何类型的列使用 NULL INDICATOR 来指示该列为 NULL 这样来表示的。 在不使用此选项的情况下，如果源列中不包含行实例的数据，那么将发生下列情况之一: • 对于 DEL/ASC/WSF 文件: 如果该列可为空，那么将装入 NULL。如果该列不可空，那么实用程序将拒绝该行。

表 21. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）

修饰符	描述
codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输入数据集中的数据的代码页。在导入操作期间，将采用此代码页的字符数据转换为采用应用程序代码页。 下列规则适用: • 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。 • nullindchar 必须指定标准 ASCII 代码集中包含的代码点 x20 与 x7F 之间（包括 x20 和 x7F 在内）的符号。这指的是 ASCII 符号和代码点。 **注**: 1. codepage 修饰符不能与 lobsinfile 修饰符一起使用。 2. 如果在将代码页从应用程序代码页转换为数据库代码页时进行了数据扩展，那么数据可能会被截断，并且有可能丢失数据。

表 21. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）(续)

修饰符	描述
dateformat="*x*"	*x* 是源文件中的数据所采用的格式[2]。有效日期元素包括: YYYY - 年份（四位数，范围是 0000 到 9999） M - 月份（一位数或两位数，范围是 1 到 12） MM - 月份（两位数，范围是 1 到 12；与 M 元素互斥） D - 日（一位数或两位数，范围是 1 到 31） DD - 日（两位数，范围是 1 到 31；与 D 元素互斥） DDD - 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥） 对于未指定的每个元素，将为它指定缺省值 1。以下是日期格式的一些示例: "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
implieddecimal	隐含的小数点所在的位置由列定义来确定；不再假定它位于值的末尾。例如，会将值 12345 作为 123.45 而不是 12345.00 装入 DECIMAL(8,2) 列。
timeformat="*x*"	*x* 是源文件中的时间格式[2]。有效时间元素包括: H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24） HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥） M - 分钟（一位数或两位数，范围是 0 到 59） MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥） S - 秒（一位数或两位数，范围是 0 到 59） SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥） SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥） TT - 正午指示符（AM 或 PM） 对于未指定的每个元素，将为它指定缺省值 0。以下是时间格式的一些示例: "HH:MM:SS" "HH.MM TT" "SSSSS"

表 21. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述

timestampformat="*x*"	*x* 是源文件中的时间戳记格式[2]。有效时间戳记元素包括:
	
	YYYY – 年份（四位数，范围是 0000 到 9999）
	M – 月份（一位数或两位数，范围是 1 到 12）
	MM – 月份（两位数，范围是 01 到 12；与 M 和 MMM 元素互斥）
	MMM – 月份（由三个不区分大小写的字母组成的月份名称缩写；与 M 和 MM 元素互斥）
	D – 日（一位数或两位数，范围是 1 到 31）
	DD – 日（两位数，范围是 1 到 31；与 D 元素互斥）
	DDD – 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥）
	H – 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24）
	HH – 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥）
	M – 分钟（一位数或两位数，范围是 0 到 59）
	MM – 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥）
	S – 秒（一位数或两位数，范围是 0 到 59）
	SS – 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥）
	SSSSS – 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥）
	UUUUUU – 微秒（六位数，范围是 000000 到 999999；与所有其他微秒元素互斥）
	UUUUU – 微秒（五位数，范围是 00000 到 99999，映射至范围 000000 到 999990；与所有其他微秒元素互斥）
	UUUU – 微秒（四位数，范围是 0000 到 9999，映射至范围 000000 到 999900；与所有其他微秒元素互斥）
	UUU – 微秒（三位数，范围是 000 到 999，映射至范围 000000 到 999000；与所有其他微秒元素互斥）
	UU – 微秒（两位数，范围是 00 到 99，映射至范围 000000 到 990000；与所有其他微秒元素互斥）
	U – 微秒（一位数，范围是 0 到 9，映射至范围 000000 到 900000；与所有其他微秒元素互斥）
	TT – 正午指示符（AM 或 PM）
	
	对于未指定的 YYYY、M、MM、D、DD 或 DDD 元素，将为它们指定缺省值 1。对于未指定的 MMM 元素，将为它指定缺省值“Jan”。对于所有其他未指定的元素，将为它们指定缺省值 0。以下是一个表示时间戳记格式的示例:
	
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	
	MMM 元素的有效值包括: 'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov' 和 'dec'。这些值都不区分大小写。
	
	以下示例说明如何将包含用户定义的日期和时间格式的数据导入到一个称为 schedule 的表中:
	
	`db2 import from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

表 21. IMPORT 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）(续)

修饰符	描述
usegraphiccodepage	如果给定了 usegraphiccodepage，那么假定导入到图形或双字节字符大对象（DBCLOB）数据字段的数据采用的是图形代码页。而假定其他数据采用的是字符代码页。图形代码页与字符代码页是相关联的。如果指定了 codepage 修饰符，那么 IMPORT 实用程序通过该修饰符来确定字符代码页；如果未指定该修饰符，那么 IMPORT 实用程序通过应用程序的代码页来确定字符代码页。 仅当要恢复的表具有图形数据时，才应将此修饰符与由删除表恢复生成的定界数据文件一起使用。 **限制** 不能对 EXPORT 实用程序创建的 DEL 文件指定 usegraphiccodepage 修饰符，这是因为这些文件中包含只使用一种代码页编码的数据。文件中的双字节字符大对象（DBCLOB）也将忽略 usegraphiccodepage 修饰符。
xmlchar	指定采用字符代码页来 XML 文档进行编码。 处理采用指定的字符代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与字符代码页相匹配，否则将拒绝包含该文档的行。注意，字符代码页就是由文件类型修饰符 codepage 指定的值；如果未指定该修饰符，那么字符代码页就是应用程序代码页。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。
xmlgraphic	指定采用指定的图形代码页来对 XML 文档进行编码。 处理采用特定的图形代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与图形代码页相匹配，否则将拒绝包含该文档的行。注意，图形代码页就是由文件类型修饰符 codepage 指定的值的图形组件；如果未指定该修饰符，那么图形代码页就是应用程序代码页的图形组件。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。 **注**: 如果对 IMPORT 命令指定了 xmlgraphic 修饰符，那么必须采用 UTF-16 代码页来对要导入的 XML 文档进行编码。否则，XML 文档可能会因产生解析错误而被拒绝，或者在将它导入到表中数据被毁坏。

表 22. IMPORT 实用程序的有效文件类型修饰符: ASC（非定界 ASCII）文件格式

修饰符	描述
nochecklengths	如果指定了 nochecklengths，那么即使源数据的列定义超过了目标表列大小，也会尝试导入每一行。如果代码页转换导致源数据缩小，那么可以成功地导入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。
nullindchar=*x*	*x* 是单个字符。将表示空值的字符更改为 *x*。*x* 的缺省值为 Y[3]。 对于 EBCDIC 数据文件，除非字符是英文字母，否则此修饰符将区分大小写。例如，如果将空指示符指定为字母 N，那么 n 也会被识别为空指示符。

表 22. IMPORT 实用程序的有效文件类型修饰符: ASC（非定界 ASCII）文件格式 (续)

修饰符	描述
reclen=*x*	*x* 是一个整数，最大值为 32 767。会读取每行的 *x* 个字符，但未使用换行符来指示行的末尾。

striptblanks	当将数据装入到一个变长字段时，将截断任何尾部空格。如果未指定此选项，那么将保留空格。
	
	在以下示例中，`striptblanks` 将导致 IMPORT 实用程序截断尾部空格:
	`db2 import from myfile.asc of asc`
	`modified by striptblanks`
	`method l (1 10, 12 15) messages msgs.txt`
	`insert into staff`
	
	不能将此选项与 `striptnulls` 同时指定。它们是互斥选项。此选项将替换过时的 `t` 选项，支持此过时选项只是为了保持与早前版本的兼容性。

striptnulls	当将数据装入到一个变长字段时，将截断任何尾部 NULL（0x00 字符）。如果未指定此选项，那么将保留 NULL。 不能将此选项与 `striptblanks` 同时指定。它们是互斥选项。此选项将替换过时的 `padwithzero` 选项，支持此过时选项只是为了保持与早前版本的兼容性。

表 23. IMPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式

修饰符	描述

chardel*x*	*x* 是单个字符串定界符。缺省值是双引号（"）。使用指定的字符而不是使用双引号将字符串引起来[34]。如果您想显式地指定双引号作为字符串定界符，那么应按如下所示指定双引号:
	`modified by chardel""`
	
	也可以指定单引号（'）作为字符串定界符。在以下示例中，`chardel''` 会导致 IMPORT 实用程序将它遇到的任何单引号（'）解释为字符串定界符:
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`method p (1, 4) insert into staff (id, years)"`

coldel*x*	*x* 是一个单字符列定界符。缺省值是逗号（,）。使用指定字符而不是逗号来表示列的末尾[34]。
	
	在以下示例中，`coldel;` 会导致 IMPORT 实用程序将它遇到的任何分号（;）解释为列定界符:
	`db2 import from myfile.del of del`
	`modified by coldel;`
	`messages msgs.txt insert into staff`

decplusblank	加号字符。导致在正的十进制值前面加上空格而不是加号（+）。缺省操作是在正的十进制值前面加上加号。

decpt*x*	*x* 是单个字符，它取代句点作为小数点字符。缺省值是句点（.）。使用指定字符而不是句点作为小数点字符[34]。
	
	在以下示例中，`decpt;` 会导致 IMPORT 实用程序将它遇到的任何分号（;）解释为小数点:
	`db2 "import from myfile.del of del`
	`modified by chardel''`
	`decpt; messages msgs.txt insert into staff"`

表 23. IMPORT 实用程序的有效文件类型修饰符: DEL（定界 ASCII）文件格式 (续)

修饰符	描述

delprioritychar	当前，定界符的缺省优先级为记录定界符、字符定界符和列定界符。此修饰符通过将定界符优先级还原为字符定界符、记录定界符和列定界符，来保护依赖于旧的优先级的现有应用程序。语法:
	`db2 import ... modified by delprioritychar ...`
	例如，用以下 DEL 数据文件作为示例:
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	如果指定了 `delprioritychar` 修饰符，那么此数据文件中将只有两行。第二个 <row delimiter> 将被解释为第二行的第一个数据列，而第一个和第三个 <row delimiter> 被解释为实际的记录定界符。如果未指定此修饰符，那么此数据文件中将有三行，每一行都用 <row delimiter> 定界。

keepblanks	保留类型为 CHAR、VARCHAR、LONG VARCHAR 或 CLOB 的每个字段中的前导空格和尾部空格。如果未指定此选项，那么会除去字符定界符外部的所有前导空格和尾部空格，并且会在表中插入 NULL 表示所有空白字段。
nochardel	IMPORT 实用程序将假定在列定界符之间找到的所有字节都是列数据的一部分。字符定界符将被解析为列数据的一部分。如果数据是使用 DB2 导出的，那么不应指定此选项（除非在导出时指定了 nochardel）。提供此修饰符的目的是支持不具有字符定界符的供应商数据文件。未正确使用此修饰符可能会导致数据丢失或毁坏。 不能将此选项与 `chardelx`、`delprioritychar` 或 `nodoubledel` 同时指定。它们是互斥选项。
nodoubledel	不识别双字符定界符。

表 24. IMPORT 实用程序的有效文件类型修饰符: IXF 文件格式

修饰符	描述
forcein	指示实用程序接受数据（即使代码页不匹配也接受），并且阻止代码页之间进行转换。 将检查定长目标字段，以验证它们对于数据来说是否足够大。如果指定了 `nochecklengths`，那么不会进行检查，并且将尝试导入每一行。
indexixf	指示实用程序删除当前已对现有表定义的所有索引，并根据 PC/IXF 文件中的索引定义来创建新的索引。仅当替换表的内容时才使用此选项。不能对视图使用此选项，当指定了 *insert-column* 时也不能指定此选项。
indexschema=*schema*	在创建索引期间，对索引名称使用指定的*模式*。如果未指定*模式*（但是指定了关键字 `indexschema`），那么使用连接用户标识。如果未指定此关键字，那么使用 IXF 文件中的模式。
nochecklengths	如果指定了 `nochecklengths`，那么即使源数据的列定义超过了目标表列大小，也会尝试导入每一行。如果代码页转换导致源数据缩小，那么可以成功地导入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。
forcecreate	指定在导入操作期间返回 SQL3311N 后也应创建表，但是可能会丢失信息或者信息有限。

表 25. 同时使用 codepage 和 usegraphiccodepage 时的 IMPORT 行为

codepage=N	usegraphiccodepage	IMPORT 行为
缺少	缺少	假定文件中的所有数据都采用应用程序代码页。
存在	缺少	假定文件中的所有数据都采用代码页 N。 **警告**: 如果 N 是单字节代码页，那么将图形数据导入到数据库中时将毁坏该数据。
缺少	存在	假定文件中的字符数据都采用应用程序代码页。假定图形数据要采用应用程序图形数据的代码页。 如果应用程序代码页是单字节，那么假定所有数据都采用应用程序代码页。 **警告**: 如果应用程序代码页是单字节，那么图形数据在导入到数据库中将毁坏数据，即使数据库中包含图形列亦如此。
存在	存在	假定字符数据采用代码页 N。假定图形数据采用图形代码页 N。 如果 N 是单字节或双字节代码页，那么假定所有数据都采用代码页 N。 **警告**: 如果 N 是单字节代码页，那么将图形数据导入到数据库中时将毁坏该数据。

注:

1. 如果您尝试将不受支持的文件类型与 MODIFIED BY 选项配合使用，IMPORT 实用程序不会发出警告。如果尝试这样做，导入操作将失败，并且会返回错误代码。
2. 日期格式字符串两边必须具有双引号。字段分隔符不能包含下列任何字符: a-z、A-Z 和 0-9。字段分隔符不应与 DEL 文件格式中的字符定界符或字段定界符相同。如果元素的开始和结束位置是明确的，那么字段分隔符是可选的。如果使用了诸如 D、H、M 或 S 之类的元素（取决于修饰符），那么由于条目长度是可变的，因此可能存在不明确性。

   对于时间戳记格式，必须要注意避免月份描述符与分钟描述符之间的不明确性，这是因为它们都使用字母 M。月份字段必须与其他日期字段相邻。而分钟字段必须与其他时间字段相邻。以下是一些不明确的时间戳记格式:

   ```
   "M"（既可能是月份，也可能是分钟）
      "M:M"（无法区分哪个是月份，哪个是分钟）
      "M:YYYY:M"（两者都将被解释为月份。）
      "S:M:YYYY"（与时间值和日期值都相邻）
   ```

   在不明确的情况下，实用程序将报告一条错误消息，并且操作将失败。

   以下是一些明确的时间戳记格式:

   ```
   "M:YYYY"（表示月份）
      "S:M"（表示分钟）
      "M:YYYY:S:M"（前者表示月份，后者表示分钟）
      "M:H:YYYY:M:D"（前者表示分钟，后者表示月份）
   ```

   在某些字符（例如，双引号和反斜杠）前面必须添加转义字符（例如，@2329。

3. 为 chardel、coldel 或 decpt 文件类型修饰符指定的字符值必须采用源数据的代码页来指定。

   可以使用 xJJ 或 0xJJ 语法来指定字符代码点（而不是字符符号）。其中 JJ 是代码点的十六进制表示法。例如，要指定 # 字符作为列定界符，可使用下列方法之一:

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

4. *移动数据时的定界符注意事项*列示了可以用作定界符的字符存在的限制。
5. 在导入到昵称时，将不允许使用下列文件类型修饰符:
   - indexixf
   - indexschema
   - dldelfiletype
   - nodefaults
   - usedefaults
   - no_type_idfiletype
   - generatedignore
   - generatedmissing
   - identityignore
   - identitymissing
   - lobsinfile
6. XML 列不支持 **WSF** 文件格式。
7. XML 列不支持 CREATE 方式。
8. 所有 XML 数据必须位于与主数据文件分隔开的 XML 文件中。主数据文件中的每个 XML 列都必须存在 XML 数据说明符（XDS）（或者空值）。
9. 除非指定了 XMLCHAR 或 XMLGRAPHIC 文件类型修饰符，否则将假定 XML 文档采用 Unicode 格式或者包含一个包括编码属性的声明标记。
10. 将拒绝包含结构不当的文档的行。
11. 如果指定了 XMLVALIDATE 选项，那么在插入成功验证了其匹配模式的文档时，会使用模式信息对这些文档进行注释。如果行中包含的文档针对其匹配模式验证失败，那么将拒绝这些行。要成功执行验证，调用 IMPORT 实用程序的用户拥有的特权必须至少包括下列其中一项权限或特权:
    - SYSADM 或 DBADM 权限
    - 对验证时要使用的 XML 模式的 USAGE 特权
12. 导入到包含隐式隐藏的 ROW CHANGE TIMESTAMP 列的表中时，将不采用该列的隐式隐藏属性。因此，如果列的数据在要导入的数据中不存在，且不存在任何显式列列表，那么*必须*在导入命令中指定 rowchangetimestampmissing 文件类型修饰符。

db2Import - 将数据导入表、层次结构、昵称或视图中

将具有受支持文件格式的外部文件中的数据插入到表、层次结构、昵称或视图中。LOAD 实用程序比此功能的导入速度更快。但是，LOAD 实用程序不支持在层次结构级别装入数据或者将数据装入昵称中。

权限

- 使用 INSERT 选项进行 IMPORT（导入）时需要具有下列权限之一:
  - sysadm
  - dbadm
  - 对参与的每个表、视图或昵称的 CONTROL 特权
  - 对参与的每个表或视图的 INSERT 和 SELECT 特权
- 使用 INSERT_UPDATE 选项 IMPORT（导入）到现有表时需要具有下列权限之一:
  - sysadm
  - dbadm
  - 对表、视图或昵称的 CONTROL 特权
  - 对参与的每个表或视图的 INSERT、SELECT、UPDATE 和 DELETE 特权
- 使用 REPLACE 或 REPLACE_CREATE 选项 IMPORT（导入）到现有表时需要具有下列权限之一:
  - sysadm
  - dbadm
  - 对表或视图的 CONTROL 特权
  - 对表或视图的 INSERT、SELECT 和 DELETE 特权
- 使用 CREATE 或 REPLACE_CREATE 选项 IMPORT（导入）到新表时需要具有下列权限之一:
  - sysadm
  - dbadm
  - 对数据库的 CREATETAB 权限、对表空间的 USE 特权以及下列其中一项权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果该表的隐式或显式模式名称不存在的话）
    - 对模式的 CREATEIN 特权（如果该表的模式名称引用现有模式的话）
- 使用 CREATE 或 REPLACE_CREATE 选项 IMPORT（导入）到不存在的表或层次结构时需要具有下列权限之一:
  - sysadm
  - dbadm
  - 对数据库的 CREATETAB 权限以及下列其中一项权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果表的模式名称不存在的话）
    - 对模式的 CREATEIN 特权（如果表的模式存在的话）
    - 对层次结构中的每个子表的 CONTROL 特权（如果使用了整个层次结构上的 REPLACE_CREATE 选项的话）
- 使用 REPLACE 选项 IMPORT（导入）到现有层次结构时需要具有下列权限之一:
  - sysadm
  - dbadm
  - 对层次结构中每个子表的 CONTROL 特权

必需的连接

数据库。如果启用了隐式连接，那么将建立与缺省数据库的连接。

API 包含文件

db2ApiDf.h

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
    db2Import (
      db2Uint32 versionNumber,
      void * pParmStruct,
      struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ImportStruct
{
      char *piDataFileName;
      struct sqlu_media_list *piLobPathList;
      struct sqldcol *piDataDescriptor;
      struct sqlchar *piActionString;
      char *piFileType;
      struct sqlchar *piFileTypeMod;
      char *piMsgFileName;
      db2int16 iCallerAction;
      struct db2ImportIn *piImportInfoIn;
      struct db2ImportOut *poImportInfoOut;
      db2int32 *piNullIndicators;
   struct sqllob *piLongActionString;
} db2ImportStruct;

typedef SQL_STRUCTURE db2ImportIn
{
      db2Uint64 iRowcount;
      db2Uint64 iRestartcount;
      db2Uint64 iSkipcount;
      db2int32 *piCommitcount;
      db2Uint32 iWarningcount;
      db2Uint16 iNoTimeout;
      db2Uint16 iAccessLevel;
      db2Uint16 *piXmlParse;
      struct db2DMUXmlValidate *piXmlValidate;
} db2ImportIn;

typedef SQL_STRUCTURE db2ImportOut
{
      db2Uint64 oRowsRead;
      db2Uint64 oRowsSkipped;
      db2Uint64 oRowsInserted;
      db2Uint64 oRowsUpdated;
      db2Uint64 oRowsRejected;
      db2Uint64 oRowsCommitted;
} db2ImportOut;

typedef SQL_STRUCTURE db2DMUXmlMapSchema
{
      struct db2Char                      iMapFromSchema;
      struct db2Char                      iMapToSchema;
} db2DMUXmlMapSchema;

typedef SQL_STRUCTURE db2DMUXmlValidateXds
{
      struct db2Char *piDefaultSchema;
      db2Uint32 iNumIgnoreSchemas;
      struct db2Char *piIgnoreSchemas;
      db2Uint32 iNumMapSchemas;
```

```
      struct db2DMUXmlMapSchema *piMapSchemas;
} db2DMUXmlValidateXds;

typedef SQL_STRUCTURE db2DMUXmlValidateSchema
{
      struct db2Char *piSchema;
} db2DMUXmlValidateSchema;

typedef SQL_STRUCTURE db2DMUXmlValidate
{
      db2Uint16 iUsing;
      struct db2DMUXmlValidateXds *piXdsArgs;
      struct db2DMUXmlValidateSchema *piSchemaArgs;
} db2DMUXmlValidate;

SQL_API_RC SQL_API_FN
    db2gImport (
      db2Uint32 versionNumber,
      void * pParmStruct,
      struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gImportStruct
{
      char *piDataFileName;
      struct sqlu_media_list *piLobPathList;
      struct sqldcol *piDataDescriptor;
      struct sqlchar *piActionString;
      char *piFileType;
      struct sqlchar *piFileTypeMod;
      char *piMsgFileName;
      db2int16 iCallerAction;
      struct db2gImportIn *piImportInfoIn;
      struct dbg2ImportOut *poImportInfoOut;
      db2int32 *piNullIndicators;
      db2Uint16 iDataFileNameLen;
      db2Uint16 iFileTypeLen;
      db2Uint16 iMsgFileNameLen;
   struct sqllob *piLongActionString;
} db2gImportStruct;

typedef SQL_STRUCTURE db2gImportIn
{
      db2Uint64 iRowcount;
      db2Uint64 iRestartcount;
      db2Uint64 iSkipcount;
      db2int32 *piCommitcount;
      db2Uint32 iWarningcount;
      db2Uint16 iNoTimeout;
      db2Uint16 iAccessLevel;
      db2Uint16 *piXmlParse;
      struct db2DMUXmlValidate *piXmlValidate;
} db2gImportIn;

typedef SQL_STRUCTURE db2gImportOut
{
      db2Uint64 oRowsRead;
      db2Uint64 oRowsSkipped;
      db2Uint64 oRowsInserted;
      db2Uint64 oRowsUpdated;
      db2Uint64 oRowsRejected;
      db2Uint64 oRowsCommitted;
} db2gImportOut;
```

db2Import API 参数

versionNumber

输入。指定作为第二个参数 pParmStruct 传入的结构的版本和发行版级别。

pParmStruct

输入/输出。指向 db2ImportStruct 结构的指针。

pSqlca

输出。指向 sqlca 结构的指针。

db2ImportStruct 数据结构参数

piDataFileName

输入。一个字符串，它包含要从其中导入数据的外部输入文件的路径和名称。

piLobPathList

输入。指向 sqlu_media_list 的指针，它的 media_type 字段设置为 SQLU_LOCAL_MEDIA，并且它的 sqlu_media_entry 结构列示可以找到 LOB 文件的客户机上的路径。当导入到昵称时，此参数无效。

piDataDescriptor

输入。指向 sqldcol 结构的指针，该结构中包含有关所选择的要从外部文件导入的列的信息。dcolmeth 字段的值确定 IMPORT 实用程序如何解释此参数中提供的其余信息。此参数的有效值为:

SQL_METH_N

名称。按列名从外部输入文件中选择列。

SQL_METH_P

位置。按列位置从外部输入文件中选择列。

SQL_METH_L

位置。按列位置从外部输入文件中选择列。对于因存在下列任何一种情况而变得无效的位置对，数据库管理器将拒绝具有这样的位置对的导入调用:

- 开始位置或结束位置不在 1 到带符号的最大两字节整数范围内。
- 结束位置小于开始位置。
- 由位置对定义的输入列宽度与目标列的类型和长度不兼容。

两个位置都等于零的位置对表示要为可空列填充 NULL。

SQL_METH_D

缺省值。如果 piDataDescriptor 为 NULL 或者设置为 SQL_METH_D，那么已从外部输入文件中缺省选择了列。在这种情况下，列数和列规范数组都将被忽略。对于 DEL、IXF 或 WSF 文件，外部输入文件中前 n 列数据采用它们的自然顺序。其中 n 是要将数据导入到的数据库列的列数。

piActionString

不推荐使用。替换为 piLongActionString。

piLongActionString

输入。指向 *sqllob* 结构的指针，该结构包含长度为 4 字节的字段，后跟用来指定会影响表的操作的字符数组。

字符数组的格式如下所示:

```
{INSERT | INSERT_UPDATE | REPLACE | CREATE | REPLACE_CREATE}
INTO {tname[(tcolumn-list)] |
[{ALL TABLES | (tname[(tcolumn-list)][, tname[(tcolumn-list)]])}]
[IN] HIERARCHY {STARTING tname | (tname[, tname])}
[UNDER sub-table-name | AS ROOT TABLE]}
```

INSERT

将已导入的数据添加至表，但不更改现有表数据。

INSERT_UPDATE

如果已导入的行的主键值不在表中，那么添加这些行；如果找到了它们的主键值，那么将它们用于更新。仅当目标表具有主键，并且指定的（或隐含的）要导入的目标列的列表中包含该主键的所有列时，此选项才有效。此选项不能应用于视图。

REPLACE

通过截断表对象来删除表中现有的所有数据，然后插入已导入的数据。表定义和索引定义不会发生更改。（如果 indexixf 在 FileTypeMod 中，并且 FileType 为 SQL_IXF，那么会删除和替换索引。）如果尚未定义表，那么将返回错误。

注: 如果在删除现有数据后发生了错误，那么该数据会丢失。
当导入到昵称时，此参数无效。

CREATE

注: 不推荐使用 CREATE 参数，将来的发行版中可能会将其除去。有关其他详细信息，请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”。

如果未定义指定的表，那么使用指定的 PC/IXF 文件中的信息来创建表定义和行内容。如果先前由 DB2 导出了文件，那么也会创建索引。如果已经定义了指定的表，那么将返回错误。此选项仅对于 PC/IXF 文件格式有效。当导入到昵称时，此参数无效。

REPLACE_CREATE

注: 不推荐使用 REPLACE_CREATE 参数，将来的发行版中可能会将其除去。有关其他详细信息，请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”。

如果已经定义了指定的表，那么使用 PC/IXF 文件中的 PC/IXF 行信息来替换表内容。如果尚未定义表，那么使用指定的 PC/IXF 文件中的信息来创建表定义和行内容。如果先前由 DB2 导出了 PC/IXF 文件，那么也会创建索引。此选项仅对于 PC/IXF 文件格式有效。

注: 如果在删除现有数据后发生了错误，那么该数据会丢失。
当导入到昵称时，此参数无效。

tname 要将数据插入到的表、类型表、视图或对象视图的名称。如果应该指定限定名称或者非限定名称，那么除了使用早期服务器的情况以外，都可以为 REPLACE、INSERT_UPDATE 或 INSERT 指定别名。如果它是一个视图，那么不能是只读视图。

tcolumn-list

要将数据插入到的表或视图列的列名列表。列名之间必须用逗号分

隔。如果未指定列名，那么使用 CREATE TABLE 或 ALTER TABLE 语句中所定义的列名。如果没有为类型表指定列的列表，那么会将数据插入到每个子表中的所有列。

sub-table-name
当使用 CREATE 选项创建一个或多个子表时指定父表。

ALL TABLES
这是仅用于层次结构的隐式关键字。当导入层次结构时，缺省情况是导入在 traversal-order-list 中指定的所有表。

HIERARCHY
指定要导入分层数据。

STARTING
仅用于层次结构的关键字。指定要使用缺省顺序，即，从给定的子表名开始。

UNDER
仅用于层次结构和 CREATE 的关键字。指定要在给定的子表下创建新的层次结构、子层次结构或子表。

AS ROOT TABLE
仅用于层次结构和 CREATE 的关键字。指定将创建新的层次结构、子层次结构或子表作为独立层次结构。

tname 和 tcolumn-list 参数分别对应于 SQL INSERT 语句的 tablename 和 colname 列表，并且具有相同的限制。

tcolumn-list 中的列与外部列（指定的或者隐含的）是根据它们在列表或结构中的位置来进行匹配的（在 sqldcol 结构中指定的第一列中的数据，被插入到与 tcolumn-list 的第一个元素相对应的表或视图字段中）。

如果指定的列数不相同，那么实际处理的列数将是较小的列数。这将导致出现错误（因为不可空的表字段中没有放入任何值）或参考消息（因为忽略了某些外部文件列）。

当导入到昵称时，此参数无效。

piFileType
输入。一个字符串，用来指示外部文件中数据的格式。受支持的外部文件格式包括:

SQL_ASC
非定界 ASCII。

SQL_DEL
定界 ASCII，用于与 dBase、BASIC 和 IBM Personal Decision Series 程序以及许多其他数据库管理器和文件管理器进行交换。

SQL_IXF
PC 版本的集成交换格式，这是用于导出表中的数据，以便稍后可将该数据导入同一个表或者导入另一个数据库管理器表的首选方法。

SQL_WSF
用于与 Lotus Symphony 和 1-2-3 程序进行交换的工作表格式。当导入到昵称时，WSF 文件类型不受支持。

piFileTypeMod

输入。指向某一结构的指针，该结构包含一个两字节长的字段，后跟用来指定一个或多个处理选项的字符数组。如果此指针为 NULL，或者指向的结构未包含任何字符，那么会将此操作解释为选择缺省规范。

并不是所有选项都可用于所有受支持的文件类型。请参阅下面的相关链接: "IMPORT 实用程序的文件类型修饰符"。

piMsgFileName

输入。一个字符串，它包含由实用程序返回的错误消息、警告消息和参考消息的目标。它可以是操作系统文件或标准设备的路径和名称。如果该文件已存在，那么会将信息追加至它。如果它不存在，那么会创建文件。

iCallerAction

输入。调用者请求的操作。有效值为:

SQLU_INITIAL

初始调用。必须在首次调用 API 时使用此值。如果初始调用或任何后续调用已返回，但是要求进行调用的应用程序执行某些操作后才能完成所请求的导入操作，那么必须将调用者操作设置为下列其中一项:

SQLU_CONTINUE

继续处理。在初始调用返回的结果是实用程序要求用户输入（例如，要求对磁带条件结束作出响应）后，只能在对 API 的后续调用中使用此值。它指定实用程序所请求的用户操作已完成，该实用程序可以继续处理初始请求。

SQLU_TERMINATE

终止处理。在初始调用返回的结果是实用程序要求用户输入（例如，要求对磁带条件结束作出响应）后，只能在对 API 的后续调用中使用此值。它指定未执行实用程序所请求的用户操作，并且实用程序将终止处理初始请求。

piImportInfoIn

输入。指向 db2ImportIn 结构的指针。

poImportInfoOut

输出。指向 db2ImportOut 结构的指针。

piNullIndicators

输入。仅适用于 ASC 文件。用来指示列数据是否可空的整数数组。此数组中的元素数必须与输入文件中的列数相匹配；此数组中的元素与从数据文件中导入的列之间存在一一对应关系。因此，元素的数目必须与 piDataDescriptor 参数的 dcolnum 字段的值相等。该数组中的每个元素都包含一个数字，用来标识数据文件中要用作空指示符字段的列。该元素也可以是零，指示表列是不可空的。如果元素不为零，那么数据文件中所标识的列必须包含 Y 或 N。如果为 Y，那么表示表列数据是 NULL；如果为 N，那么表示表列数据不是 NULL。

piXmlPathList

输入。指向 sqlu_media_list 的指针，它的 media_type 字段设置为 SQLU_LOCAL_MEDIA，并且它的 sqlu_media_entry 结构列示客户机上用于存储 XML 文件的路径。

db2ImportIn 数据结构参数

iRowcount

输入。要装入的物理记录数。允许用户只装入文件中前面的 iRowcount 行。如果 iRowcount 为 0，那么 IMPORT 实用程序将尝试处理文件中的所有行。

iRestartcount

输入。在开始插入或更新记录前要跳过的记录数。此参数与 iSkipcount 参数具有同等的功能。iRestartcount 与 iSkipcount 参数是互斥的。

iSkipcount

输入。在开始插入或更新记录前要跳过的记录数。此参数与 iRestartcount 参数具有同等的功能。

piCommitcount

输入。在将记录落实到数据库前要导入的记录数。每当导入了 piCommitcount 条记录时就会执行落实。如果值为 NULL，那么会指定缺省落实计数值。对于脱机导入，值为零；而对于联机导入，值为 AUTOMATIC。落实计数 AUTOMATIC 是通过传入 DB2IMPORT_COMMIT_AUTO 值来指定的。

iWarningcount

输入。在发出 iWarningcount 个警告后就停止导入操作。如果希望不产生警告，那么设置此参数，但是需要验证使用的是正确的文件和表。如果指定了不正确的导入文件或目标表，那么 IMPORT 实用程序将对它试图导入的每一行生成一个警告，这将导致导入失败。

如果 iWarningcount 为 0，或者未指定此选项，那么无论发出多少个警告，都将继续执行导入操作。

iNoTimeout

输入。指定 IMPORT 实用程序在等待锁定时不会超时。此选项取代了数据库配置参数 locktimeout。其他应用程序将不受影响。有效值为:

DB2IMPORT_LOCKTIMEOUT

指示要考虑 locktimeout 配置参数的值。

DB2IMPORT_NO_LOCKTIMEOUT

指示不会超时。

iAccessLevel

输入。指定访问级别。有效值为:

- SQLU_ALLOW_NO_ACCESS

指定 IMPORT 实用程序将互斥锁定表。

- SQLU_ALLOW_WRITE_ACCESS

指定在导入过程中阅读器和写程序仍然可以访问表中的数据。

插入第一行时，将获得对目标表的意向互斥（IX）锁定。这允许并发阅读器和写程序访问表数据。联机方式与 REPLACE、CREATE 或 REPLACE_CREATE 导入选项不兼容。不支持将联机方式与缓冲插入一起使用。导入操作将定期落实已插入的数据，以防止锁定升级为表锁定，并且避免用尽活动日志空间。即使未使用 piCommitCount 参数，也将执行这些落实操作。在每次落实时，导入操作都将失去它的 IX 表锁定，但是在落实后又将尝试重新获取锁定。当导入到昵称时，此参数是必需的，并且必须为 piCommitCount 参数指定一个有效值（AUTOMATIC 不是有效选项）。

piXmlParse

输入。应该对 XML 文档执行的解析类型。在包含目录中的 db2ApiDf 头文件中找到的有效值为:

DB2DMU_XMLPARSE_PRESERVE_WS
应该保留空格。

DB2DMU_XMLPARSE_STRIP_WS
应该去掉空格。

piXmlValidate

输入。指向 db2DMUXmlValidate 结构的指针。指示应该对 XML 文档执行 XML 模式验证。

db2ImportOut 数据结构参数

oRowsRead

输出。在导入期间从文件中读取的记录数。

oRowsSkipped

输出。在开始插入或更新前跳过的记录数。

oRowsInserted

输出。已插入到目标表中的行数。

oRowsUpdated

输出。使用已导入的记录（也就是其主键值在表中已存在的记录）中的信息更新的目标表中的行数。

oRowsRejected

输出。未能导入的记录数。

oRowsCommitted

输出。已成功导入并且已落实到数据库的记录数。

db2DMUXmlMapSchema 数据结构参数

iMapFromSchema

输入。要映射自的 XML 模式的 SQL 标识。

iMapToSchema

输入。要映射至的 XML 模式的 SQL 标识。

db2DMUXmlValidateXds 数据结构参数

piDefaultSchema

输入。当 XDS 不包含 SCH 属性时，应该用于验证的 XML 模式的 SQL 标识。

iNumIgnoreSchemas

输入。如果 XDS 中的 SCH 属性引用了 XML 模式，那么此参数是指在验证 XML 模式期间将忽略的 XML 模式的数目。

piIgnoreSchemas

输入。如果 XDS 中的 SCH 属性引用了 XML 模式，那么此参数是指在验证 XML 模式期间将忽略的 XML 模式的列表。

iNumMapSchemas

输入。在验证 XML 模式期间将映射的 XML 模式的数目。模式映射对中的第一个模式表示 XDS 中的 SCH 属性引用的模式。模式对中的第二个模式表示应该用来执行模式验证的模式。

piMapSchemas

输入。XML 模式对的列表，每个 XML 模式对都表示一个模式与另一个模式之间的映射。模式对中的第一个模式表示 XDS 中的 SCH 属性引用的模式。模式对中的第二个模式表示应该用来执行模式验证的模式。

db2DMUXmlValidateSchema 数据结构参数

piSchema

输入。要使用的 XML 模式的 SQL 标识。

db2DMUXmlValidate 数据结构参数

iUsing

输入。一种规范，它指定使用哪些资源来执行 XML 模式验证。在包含目录中的 db2ApiDf 头文件中找到的有效值为:

- **DB2DMU_XMLVAL_XDS**

  应该根据 XDS 来执行验证。这相当于 CLP“XMLVALIDATE USING XDS”子句。

- **DB2DMU_XMLVAL_SCHEMA**

  应该根据指定的模式来执行验证。这相当于 CLP“XMLVALIDATE USING SCHEMA”子句。

- **DB2DMU_XMLVAL_SCHEMALOC_HINTS**

  应该根据在 XML 文档中找到的 schemaLocation 提示来执行验证。这相当于“XMLVALIDATE USING SCHEMALOCATION HINTS”子句。

piXdsArgs

输入。指向 db2DMUXmlValidateXds 结构的指针，表示相当于 CLP“XMLVALIDATE USING XDS”子句的参数。

仅当同一结构中的 iUsing 参数设置为 DB2DMU_XMLVAL_XDS 时，此参数才适用。

piSchemaArgs

输入。指向 db2DMUXmlValidateSchema 结构的指针，表示相当于 CLP“XMLVALIDATE USING SCHEMA”子句的参数。

仅当同一结构中的 iUsing 参数设置为 DB2DMU_XMLVAL_SCHEMA 时，此参数才适用。

特定于 db2gImportStruct 数据结构的参数

iDataFileNameLen

输入。指定 piDataFileName 参数的长度（以字节计）。

iFileTypeLen

输入。指定 piFileType 参数的长度（以字节计）。

iMsgFileNameLen

输入。指定 piMsgFileName 参数的长度（以字节计）。

使用说明

在开始执行导入操作前，必须采用下面任一方法来完成所有表操作并释放所有锁定:

- 关闭所有已打开的使用 WITH HOLD 子句定义的游标，并通过执行 COMMIT 语句来落实数据更改。
- 通过执行 ROLLBACK 语句来回滚数据更改。

IMPORT 实用程序使用 SQL INSERT 语句将一些行添加至目标表。

该实用程序将对输入文件中的每行数据发出一个 INSERT 语句。如果 INSERT 语句失败，那么将导致执行以下两个操作之一:

- 如果后续 INSERT 语句可能会成功，那么会将警告消息写入消息文件中，但是将继续进行处理。
- 如果后续 INSERT 语句可能会失败，并且数据库可能会被破坏，那么会将错误消息写入消息文件并停止处理。

在执行 REPLACE 或 REPLACE_CREATE 操作期间删除了旧的行后，该实用程序将执行自动落实（COMMIT）。因此，如果系统失败，或者在截断表对象后应用程序中断了数据库管理器，那么所有旧数据都会丢失。确保在使用这些选项前不再需要旧数据。

如果在执行 CREATE、REPLACE 或 REPLACE_CREATE 操作期间日志已满，那么该实用程序将对已插入的记录执行自动落实（COMMIT）。如果系统失败，或者在自动落实（COMMIT）后应用程序中断了数据库管理器，那么具有部分数据的表将保留在数据库中。使用 REPLACE 或 REPLACE_CREATE 选项来重新运行整个导入操作，或者在 iRestartcount 参数设置为已成功导入的行数的情况下执行插入（INSERT）操作。

缺省情况下，不会对 INSERT 或 INSERT_UPDATE 选项执行自动落实（COMMIT）。但是，如果 *piCommitcount 参数不为零，那么将执行自动落实（COMMIT）。当日志已满时就会导致回滚（ROLLBACK）。

每当 IMPORT 实用程序执行落实（COMMIT）时，会将两条消息写入消息文件中: 一条消息指示要落实的记录数; 在成功执行落实（COMMIT）后将写入另一条消息。当在失败后重新启动导入操作时，请指定要跳过的记录数，此数目是在上一次成功执行落实（COMMIT）时确定的。

IMPORT 实用程序将接受带有次要的不兼容问题的输入数据（例如，可通过使用填充或截断来导入的字符数据，以及可以使用另一种数字数据类型来导入的数字数据），但是不会接受具有主要的不兼容问题的数据。

如果一个对象表具有除它本身之外的任何从属项，或者如果对象视图的基本表具有任何从属项（包括它本身），那么不能 REPLACE 或 REPLACE_CREATE 该对象表或对象视图。要替换这样的一个表或视图，请执行以下操作:

1. 删除该表是父表的所有外键。
2. 运行 IMPORT 实用程序。
3. 改变该表以重新创建外键。

如果在重新创建外键时发生了错误，那么修改数据以保持引用完整性。

根据 PC/IXF 文件创建表时，不会保留引用约束和外键定义。（如果先前是使用 SELECT * 来导出数据的，那么将保留主键定义。）

在导入到远程数据库时，要求服务器上有足够的磁盘空间来复制输入数据文件、输出消息文件以及满足数据库可能增大的需求。

如果对远程数据库运行导入操作，并且输出消息文件很长（超过 60 KB），那么返回给客户机上的用户的消息文件可能会在导入操作过程中丢失消息。但始终会保留前 30 KB 的消息信息和最后 30 KB 的消息信息。

如果使用 piDataDescriptor 的非缺省值，或者在 piLongActionString 中指定表列的显式列表，那么会使导入到远程数据库中时的速度更慢。

必须存在数据库表或层次结构，才能导入采用 ASC、DEL 或 WSF 文件格式的数据；但是，如果表尚不存在，那么 IMPORT CREATE 或 IMPORT REPLACE_CREATE 在从 PC/IXF 文件中导入数据时将创建表。对于类型表，IMPORT CREATE 可以创建类型层次结构以及表层次结构。

应该使用 PC/IXF 导入来在数据库之间移动数据（包括分层数据）。如果包含行分隔符的字符数据已导出到定界 ASCII（DEL）文件并且已由文本传输程序处理，那么包含行分隔符的字段将收缩或展开。

默认 ASC 和 DEL 文件中的数据采用的是执行导入操作的客户机应用程序的代码页。当导入采用其他代码页的数据时，建议使用 PC/IXF 文件（这些文件允许使用其他代码页）。如果 PC/IXF 文件与 IMPORT 实用程序采用相同的代码页，那么像常规应用程序一样进行处理。如果它们使用不同的代码页，并且指定了 FORCEIN 选项，那么 IMPORT 实用程序将认为 PC/IXF 文件中的数据与执行导入操作的应用程序具有相同的代码页。即使这两种代码页之间存在转换表，也会发生这种情况。如果它们使用不同的代码页，并且未指定 FORCEIN 选项，代码页之间还存在转换表，那么 PC/IXF 文件中的所有数据都将从文件代码页转换为应用程序代码页。如果它们使用不同的代码页，并且未指定 FORCEIN 选项，代码页之间不存在转换表，那么导入操作将失败。这仅适用于 DB2 AIX 版客户机上的 PC/IXF 文件。

对于 8KB 页（接近 1012 列这一限制）上的表对象，导入 PC/IXF 数据文件可能会导致 DB2 返回错误，这是因为已经超过了 SQL 语句的最大大小。仅当列类型为 CHAR、VARCHAR 或 CLOB 时才会发生这种情况。该限制并不适用于导入 DEL 或 ASC 文件这种情况。

可以使用 DB2 Connect 将数据导入到 DRDA 服务器（例如，DB2 OS/390 版、DB2 VM 和 VSE 版以及 DB2 OS/400 版）中。仅支持 PC/IXF 导入（INSERT 选项）。还支持 restartcnt 参数，但不支持 commitcnt 参数。

当对类型表使用 CREATE 选项时，将创建在 PC/IXF 文件中定义的每个子表；不能改变子表定义。当对类型表使用除 CREATE 之外的其他选项时，遍历顺序列表允许用户指定遍历顺序；因此，遍历顺序列表必须与在执行导出操作期间所使用的遍历顺序列表相匹配。对于 PC/IXF 文件格式，用户只需要指定目标子表名，并使用文件中存储的遍历顺序。可以使用 IMPORT 实用程序来恢复先前已导出到 PC/IXF 文件的表。该表将返回到导出它时所处的状态。

不能将数据导入到系统表、已声明临时表或总结表。

不能通过 import 实用程序创建视图。

在 Windows 操作系统上:

- 不支持导入按逻辑划分的 PC/IXF 文件。
- 不支持导入错误格式的 PC/IXF 或 WSF 文件。

联合注意事项

当使用 db2Import API 和 INSERT、UPDATE 或 INSERT_UPDATE 参数时，必须确保您对参与的昵称具有 CONTROL 特权。必须确保您在执行导入操作时想使用的昵称已存在。

导入会话 - CLP 示例

示例 1

以下示例说明如何将 myfile.ixf 中的信息导入到 STAFF 表中:

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff

SQL3150N  PC/IXF 文件中的 H 记录具有产品"DB2    01.00"、日期
"19970220" 和时间"140848"。

SQL3153N  PC/IXF 文件中的 T 记录具有名称"myfile"、
限定符"        " 和源"            "。

SQL3109N  实用程序开始装入文件"myfile"中的数据。

SQL3110N  实用程序已完成处理。从输入文件读取了"58"行。


SQL3221W  ...开始 COMMIT WORK。输入记录计数 ="58"。

SQL3222W  ...COMMIT 任何数据库更改成功。

SQL3149N  处理了输入文件中的"58"行。已在表中成功插入了"58"行。"0"行被拒绝。
```

示例 2

以下示例显示如何导入到具有标识列的表中:

TABLE1 有 4 列:

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 与 TABLE1 相同，但 C2 是 GENERATED ALWAYS 标识列。

DATAFILE1 中的数据记录（DEL 格式）:

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2 中的数据记录（DEL 格式）:

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

由于在 DATAFILE1 中没有为第 1 行和第 2 行提供标识值，因此以下命令将为这两行生成标识值。但是，分别对第 3 行和第 4 行指定了用户提供的标识值 100 和 101。

```
db2 import from datafile1.del of del replace into table1
```

要将 DATAFILE1 导入到 TABLE1 中，以便为所有行生成标识值，请发出下列其中一个命令:

```
db2 import from datafile1.del of del method P(1, 3, 4)
   replace into table1 (c1, c3, c4)
db2 import from datafile1.del of del modified by identityignore
   replace into table1
```

要将 DATAFILE2 导入到 TABLE1 中，以便为每一行生成标识值，请发出下列其中一个命令:

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
db2 import from datafile2.del of del modified by identitymissing
   replace into table1
```

如果将 DATAFILE1 导入到 TABLE2 中，但未使用任何与标识相关的文件类型修饰符，那么将插入第 1 行和第 2 行，但将拒绝第 3 行和第 4 行，这是因为这两行提供了它们自己的非空值，而标识列是 GENERATED ALWAYS 列。

示例 3

以下示例显示如何导入到具有空指示符的表中:

TABLE1 有 5 列:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 有 6 个元素:

- ELE1 位置 01 到 20
- ELE2 位置 21 到 22
- ELE5 位置 23 到 23
- ELE3 位置 24 到 27
- ELE4 位置 28 到 31
- ELE6 位置 32 到 32
- ELE6 位置 33 到 40

数据记录:

```
1...5....10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    wxyzN
Test data 4,5 and 6 WWN6789    Y
```

以下命令将 ASCFILE1 中的记录导入到 TABLE1 中:

```
db2 import from ascfile1 of asc
method L (1 20, 21 22, 24 27, 28 31)
null indicators (0, 0, 23, 32)
insert into table1 (col1, col5, col2, col3)
```

注:

1. 由于在输入文件中未提供 COL4，所以将使用该列的缺省值来将其插入到 TABLE1 中（定义该列时指定了 NOT NULL WITH DEFAULT）。
2. 位置 23 和 32 用来指示: 装入 TABLE1 时，对于给定的行，是否将 COL2 和 COL3 设置为 NULL。对于给定记录，如果该列的空指示符位置包含 Y，那么该列将是 NULL。如果包含 N，那么将输入记录（由 L(.....) 定义）中该列的数据位置中包含的数据值用作该行的列数据源。在此示例中，第 1 行中的任何一列都不是 NULL; 第 2 行中的 COL2 是 NULL; 第 3 行中的 COL3 是 NULL。
3. 在此示例中，将 COL1 和 COL5 的 NULL INDICATORS 指定为 0（零），表示数据不可为空。
4. 给定列的 NULL INDICATOR 可以在输入记录中的任何位置，但必须指定该位置，并且必须提供 Y 或 N 值。

第 4 章 Load 实用程序

Load 概述

load 实用程序能够高效地将大量数据移到新创建的表或者已包含数据的表中。此实用程序能够处理绝大多数数据类型，其中包括 XML、大对象（LOB）和用户定义的类型（UDT）。由于 load 实用程序直接将格式化的页写入数据库，而 IMPORT 实用程序却要执行 SQL INSERT，因此 load 实用程序的速度比 IMPORT 实用程序快。load 实用程序不会触发触发器，并且除了验证索引唯一性以外不执行引用约束检查或表约束检查。

LOAD 过程由四个不同的阶段组成（请参阅图 3）：

1. **装入**
   在装入阶段，将把数据装入到表中，必要时还将收集索引键和表统计信息。并且，将根据 LOAD 命令中的 **SAVECOUNT** 参数指定的时间间隔建立*保存点*或一致点。将生成消息以指示在保存点成功装入的输入行数。
2. **构建**
   在构建阶段，将根据装入阶段收集的索引键生成索引。在装入阶段将对索引键进行排序并收集索引统计信息（如果指定了 STATISTICS USE PROFILE 选项且概要文件指示收集索引状态）。这些统计信息与通过 RUNSTATS 命令收集的统计信息类似。
3. **删除**
   在删除阶段，将从表中除去导致唯一键或主键违例的行。如果指定了装入异常表，那么这些删除的行将存储在该表中。
4. **索引复制**
   在索引复制阶段，将索引数据从系统临时表空间复制到原始表空间。在指定了 READ ACCESS 选项的装入操作期间，仅当指定使用系统临时表空间来创建索引时，才会出现这种情况。

图 3. 装入过程的四个阶段是：装入、构建、删除和索引复制

注：调用 load 实用程序后，可以使用 LIST UTILITIES 命令来监视装入操作的进度。

装入数据时需要下列信息：

- 输入文件、命名管道或设备的路径和名称。
- 目标表的名称或别名。
- 输入源的格式。此格式可以是 DEL、ASC、PC/IXF 或 CURSOR。
- 是将输入数据追加到表中还是替换表中的现有数据。
- 消息文件名（如果通过 db2Load 应用程序编程接口（API）调用此实用程序）。

装入方式

- **INSERT**
  在此方式下，装入将输入数据追加至表，并且不对现有数据进行任何更改。
- **REPLACE**
  在此方式下，装入将删除表中的现有数据并用输入数据填充该表。
- **RESTART**
  在此方式下，已中断的装入将继续。在大多数情况下，装入将从它失败时所处的阶段继续。如果该阶段为装入阶段，那么装入将从上一个成功一致点继续。
- **TERMINATE**
  在此方式下，将回滚失败的装入操作。

可以指定的选项包括:

- 要装入的数据在客户机上（如果从连接的远程客户机调用 load 实用程序）。请注意，即使您指定 CLIENT 选项，也始终从服务器中读取 XML 和 LOB 数据。
- 数据装入方法: 列位置、列名或相对列位置。
- 实用程序建立一致点的频率。
- 要在其中插入数据的表列的名称。
- 在装入操作执行过程中，是否可以查询表中预先存在的数据。
- 装入操作在继续执行之前是应该等待其他实用程序或应用程序使用完表，还是应该强制其他应用程序释放锁定。
- 构建索引时要使用的备用系统临时表空间。
- 用于存储 LOB 的输入文件的路径和名称。

  注: load 实用程序不使用 COMPACT lob 选项。
- 消息文件名。在执行装入操作期间，可以指定要创建消息文件以包含与那些操作相关联的错误消息、警告消息和参考消息。使用 MESSAGES 参数来指定这些文件的名称。

  注:

  1. 只能在操作完成之后查看消息文件的内容。如果要在装入操作正在运行时查看装入消息，那么可以使用 LOAD QUERY 命令。
  2. 在消息文件中，每条消息都另起一行并包含 DB2 消息检索工具提供的信息。
- 正在装入的列值是否有隐式的小数点。
- 在装入表之后，实用程序是否应该修改可用空间量。
- 在装入过程中是否收集统计信息。仅当以 REPLACE 方式运行装入操作时，才支持此选项。根据为表定义的概要文件收集统计信息。必须在执行 LOAD 命令前使用 RUNSTATS 命令创建该概要文件。如果不存在该概要文件，但您指示装入操作根据该概要文件收集统计信息，装入操作就会失败并返回错误消息。

  如果对表追加数据，就不会收集统计信息。要收集有关追加的表的当前统计信息，请在装入过程完成后调用 RUNSTATS 实用程序。如果对包含唯一索引的表收集统计信息，并且在删除阶段删除了重复的键，那么不会更新统计信息来考虑删除掉的记录。如果您预计会有相当数量的重复记录，请不要在装入操作执行期间收集统计信息。而是，应该在装入过程完成后调用 RUNSTATS 实用程序。

- 是否保留所作更改的副本。如果保留更改副本，就可以启用数据库前滚恢复功能。如果对数据库禁用了前滚恢复功能（即，数据库配置参数 *logarchmeth1* 和 *logarchmeth2* 设置为 OFF），那么不支持此选项。如果未创建副本，但启用了前滚恢复功能，那么装入操作完成之后，表空间将处于“备份暂挂”状态。

  要使数据库完全可恢复，就需要进行记录。load 实用程序几乎完全不记录与装入数据相关联的信息。您可以选择创建表的已装入部分的副本，而不是进行记录。如果数据库环境允许在发生故障后进行数据库恢复，那么可以执行下列其中一项操作:

  – 显式请求创建表的已装入部分的副本。
  – 在装入操作完成后立即备份表所在的表空间。

  如果设置了数据库配置参数 *logindexbuild*，并且调用装入操作时指定了 COPY YES 可恢复性选项和 INCREMENTAL 建立索引选项，那么装入操作将记录对索引所作的所有修改。使用这些选项的好处是: 在使用此装入操作的日志记录进行前滚时，还将恢复索引（通常，除非 LOAD 使用 REBUILD 建立索引方式，否则不会恢复索引）。

  如果正在装入已包含数据的表，并且数据库不可恢复，那么在调用 load 实用程序之前，请确保已备份数据库或者所装入表的表空间，以便能够从错误中恢复。

  如果要对可恢复数据库执行一系列装入操作，那么与每次调用装入操作时都指定 COPY YES 选项相比，指定每个装入操作都不可恢复并在装入序列结束时进行备份的速度更快。可以使用 NONRECOVERABLE 选项来指定将装入事务标记为不可恢复，这样，以后不可能通过前滚操作恢复该事务。Rollforward 实用程序将跳过该事务，并将装入数据的表标记为“无效”。该实用程序还将忽略该对表执行的任何后续事务。在前滚操作完成后，只能废弃这样的表（请参阅图 4）。如果指定了此选项，在装入操作完成后就不会将表空间置于“备份暂挂”状态，并且在装入操作执行期间不必创建所装入数据的副本。

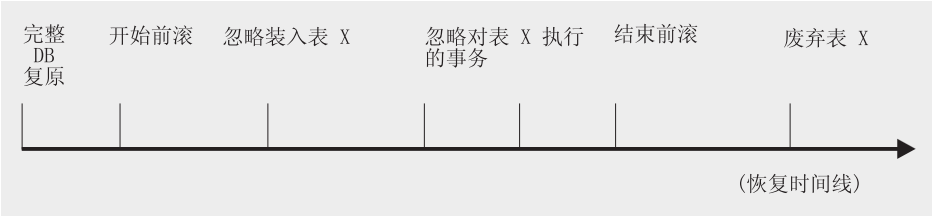


图 4. 前滚操作执行期间的不可恢复处理

- 装入操作执行期间创建临时文件时要使用的标准路径。此名称由 LOAD 命令的 TEMPFILES PATH 参数指定。缺省值是数据库路径。此路径在服务器上，并且由 DB2 实例以独占方式访问。因此，对此参数指定的任何路径名限定都必须反映服务器（而不是客户机）的目录结构，并且 DB2 实例所有者对该路径必须具有读写许可权。

使用 LOAD 所需的特权和权限

特权使用户能够创建或访问数据库资源。权限级别提供了对特权、较高级别数据库管理器维护和实用程序操作进行分组的方法。这两者一起用于控制对数据库管理器及其数据库对象的访问。用户只能访问那些他们具有相应授权（即必需的特权或权限）的对象。

要将数据装入到表中，您必须拥有下列其中一项权限或特权:

- SYSADM 权限
- DBADM 权限
- 对数据库的 LOAD 权限，以及下列特权
  - 以 INSERT 方式、TERMINATE 方式（用于终止先前的装入插入操作）或 RESTART 方式（用于重新启动先前的装入插入操作）调用 load 实用程序时，对表的 INSERT 特权
  - 以 REPLACE 方式、TERMINATE 方式（用于终止先前的装入替换操作）或 RESTART 方式（用于重新启动先前的装入替换操作）调用 load 实用程序时，对表的 INSERT 和 DELETE 特权
  - 作为装入操作使用异常表时，对异常表的 INSERT 特权
  - 对 SYSCAT.TABLES 的 SELECT 特权在 LOAD 查询目录表的某些情况下是必需的。

由于所有装入进程（通常还包括所有 DB2 服务器进程）都由实例所有者拥有，并且所有这些进程都使用实例所有者的标识来访问所需的文件，因此，实例所有者必须对输入数据文件具有读访问权。无论谁调用该命令，实例所有者都必须能够读取这些输入数据文件。

如果指定了 REPLACE 选项，那么会话授权标识必须有权废弃该表。

在 DB2 作为 Windows 服务运行的 Windows 和 Windows.NET 操作系统上，如果要从位于网络驱动器上的文件中装入数据，那么必须将 DB2 服务配置为以对这些文件具有读访问权的用户帐户下运行。

注:

- 要将数据装入到包含受保护列的表中，会话授权标识必须拥有允许对该表中所有受保护列执行写访问的 LBAC 凭证。
- 要将数据装入到包含受保护行的表中，必须已将保护该表的安全策略所包含的写访问安全标号授予会话授权标识。

LOAD 权限

在数据库级具有 LOAD 权限以及对表具有 INSERT 特权的用户可以使用 LOAD 命令来将数据装入到表中。

如果先前的装入操作是用来装入插入数据的操作，那么在数据库级具有 LOAD 权限且对表具有 INSERT 特权的用户可以执行 LOAD RESTART 或 LOAD TERMINATE 操作。

在数据库级具有 LOAD 权限同时对表具有 INSERT 和 DELETE 特权的用户可以使用 LOAD REPLACE 命令。

如果先前的装入操作是装入替换，那么还必须对该用户授予 DELETE 特权，该用户才能执行 LOAD RESTART 或 LOAD TERMINATE 操作。

如果将异常表用作装入操作的一部分，那么用户对异常表必须具有 INSERT 特权。

具有此权限的用户可以执行 QUIESCE TABLESPACES FOR TABLE、RUNSTATS 和 LIST TABLESPACES 命令。

装入数据

load 实用程序能够高效地将大量数据移到新创建的表或者已包含数据的表中。

在调用 load 实用程序前，您必须连接至（或者能够隐式地连接至）要装入数据的数据库。由于该实用程序将发出 COMMIT 语句，所以应该通过发出 COMMIT 或 ROLLBACK 语句来完成所有事务并释放所有锁定，然后再调用 load 实用程序。数据是按照出现在输入文件中的顺序装入的，但使用多维集群（MDC）表、分区表或 anyorder 文件类型修饰符时除外。如果期望使用特定顺序，那么在尝试装入操作之前对数据排序。如果需要进行集群，那么应该在执行装入前按集群索引对数据进行排序。在将数据装入到多维集群表（MDC）中时，在执行装入操作前不需要进行排序，数据将根据 MDC 表定义进行集群。在将数据装入到分区表中时，在执行装入操作前不需要进行排序，数据将根据表定义进行分区。

以下是 load 实用程序存在的一些限制（即，此列表不全面）：

- 不支持将数据装入到昵称中。
- 不支持将数据装入到类型表或者带有结构化类型列的表中。
- 不支持将数据装入到已声明临时表中。
- 只能从服务器端读取 XML 数据；如果要从客户端读取 XML 文件，那么使用 import 实用程序。
- 不能在处于“备份暂挂”状态的表空间中创建或废弃表。
- 不能将数据装入到通过 DB2 Connect 或 DB2 版本 2 之前的服务器级别访问的数据库中。仅适用于当前版本的选项不能用于先前发行版的服务器。
- 如果 LOAD REPLACE 操作期间出错，那么表中的原始数据就会丢失。您应该保留一份输入数据以便能够重新启动装入操作。
- 不会对新装入的行激活触发器。load 实用程序不会强制实施与触发器相关联的业务规则。
- 不支持装入已加密的数据。

在装入到分区表中时，load 实用程序存在下列限制（即，此列表不全面）：

- 当分区代理程序数大于 1 时，不支持一致点。
- 不支持将数据装入到数据分区子集中的同时保持其余数据分区完全联机。
- 装入操作或设置完整性暂挂操作使用的异常表不能分区。
- 当 load 实用程序以插入方式或重新启动运行并且装入目标表具有任何已拆离的从属时，那么不能重建唯一索引。

可以通过命令行处理器（CLP）、控制中心中的“装入”向导或者 db2Load 应用程序编程接口（API）来调用 load 实用程序。

使用“装入”向导

1. 在“控制中心”中，展开对象树，直到找到“表”文件夹为止。
2. 单击“表”文件夹。所有现有表都会显示在窗口右边的窗格（内容窗格）中。
3. 在内容窗格中，右键单击想要的表，然后从弹出菜单中选择“装入”。这就打开了“装入”向导。
4. 在该向导的每一页上指定成功装入数据所必需的信息。

“装入”向导的联机帮助工具提供了有关该向导的详细信息。

通过使用 CLP 发出 LOAD 命令

以下是通过 CLP 发出的 LOAD 命令示例:

```
db2 load from stafftab.ixf of ixf messages staff.msgs
         insert into userid.staff copy yes use tsm data buffer 4000
```

在此示例中:

- 将把任何警告或错误消息放入 staff.msgs 文件。
- 所作的更改的一个副本将存储在 Tivoli® Storage Manager（TSM）中。
- 在装入操作期间将使用 4000 页的缓冲区空间。

以下是通过 CLP 发出的另一个 LOAD 命令示例:

```
db2 load from stafftab.ixf of ixf messages staff.msgs
         tempfiles path /u/myuser replace into staff
```

在此示例中:

- 将替换表数据。
- 使用 TEMPFILES PATH 参数来将 /u/myuser 指定为写入临时文件的服务器路径。

注: 这些示例对装入输入文件使用相对路径名。只允许从数据库所在数据库分区上的客户机进行的调用使用相对路径名。建议使用标准路径名。

调用 load 实用程序后，可以使用 LIST UTILITIES 命令来监视装入操作的进度。在以 INSERT 方式、REPLACE 方式或 RESTART 方式执行装入操作时，将提供详细进度监视支持。发出带有 SHOW DETAILS 选项的 LIST UTILITIES 命令来查看关于当前装入阶段的详细信息。以 TERMINATE 方式执行装入操作时，将无法获取详细信息。LIST UTILITIES 命令将仅仅显示装入终止实用程序当前正在运行。

装入操作保留唯一约束、分区表范围限制、生成列和 LBAC 安全规则。对于所有其他约束，在装入操作开始时该表将处于“设置完整性暂挂”状态。在装入操作完成后，必须使用 Set Integrity 语句来使该表脱离“设置完整性暂挂”状态。

装入 XML 数据

LOAD 实用程序可有效地将大量 XML 数据移到表中。

将数据装入到 XML 表列中时，可以使用 XML FROM 选项来指定一个或多个输入 XML 数据文件的路径。例如，要从 XML 文件“/home/user/xmlpath/xmlfile1.xml”装入数据，应使用以下命令:

```
LOAD FROM data1.del OF DEL XML FROM /home/user/xmlpath INSERT INTO USER.T1
```

定界 ASCII 输入文件“data1.del”包含 XML 数据说明符（XDS），以描述要装入的 XML 数据的位置。例如，以下 XDS 描述文件“xmldata.ext”中偏移量为 123 字节处的 XML 文档（其长度为 456 字节）:

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' />
```

针对模式验证插入的文档

XMLVALIDATE 选项允许在装入 XML 文档时针对 XML 模式验证这些文档。在以下示例中，将针对定界 ASCII 输入文件“data2.del”中的 XDS 标识的模式验证入局 XML 文档:

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLVALIDATE
     USING XDS INSERT INTO USER.T2
```

在这种情况下，XDS 包含 SCH 属性及用于验证的 XML 模式的标准 SQL 标识“S1.SCHEMA_A”:

```
<XDS FIL='xmldata.ext' OFF='123' LEN='456' SCH='S1.SCHEMA_A' />
```

指定解析选项

可以使用 XMLPARSE 选项来指定是保留还是去掉装入的 XML 文档中的空格。在以下示例中，将针对带有 SQL 标识“S2.SCHEMA_A”的模式验证所有装入的 XML 文档，并且在保留空格的情况下解析这些文档:

```
LOAD FROM data2.del OF DEL XML FROM /home/user/xmlpath XMLPARSE PRESERVE
WHITESPACE XMLVALIDATE USING SCHEMA S2.SCHEMA_A INSERT INTO USER.T1
```

分区表的装入注意事项

对目标表进行分区时，将支持所有现有装入功能，但存在以下常规限制:

- 当分区代理程序数大于 1 时，不支持一致点。
- 不支持将数据装入到数据分区子集中的同时保持其余数据分区完全联机。
- 装入操作使用的异常表不能分区。
- 当 load 实用程序以插入方式或重新启动运行并且装入目标表具有任何已拆离的从属时，那么不能重建唯一索引。
- 与装入 MDC 表相同，装入分区表时将不会保留输入数据记录的精确排序。只有在单元或数据分区中才保留排序。
- 在每个数据库分区上利用多个格式化程序的装入操作仅保留输入记录的大致排序。在每个数据库分区上运行单个格式化程序，将输入记录按单元或表分区键进行分组。要在每个数据库分区上运行单个格式化程序，应显式请求 CPU_PARALLELISM 为 1。

一般装入行为

load 实用程序将数据记录插入到正确的数据分区中。在装入之前，不需要使用外部实用程序（如分割程序）来对输入数据进行分区。

load 实用程序不访问任何拆离的或相连的数据分区。数据仅插入到可视数据分区中。可视数据分区不会拆离，也不会相连。此外，装入替换操作不会截断拆离或相连的数据分区。因为 load 实用程序获取针对目录系统表的锁定，所以它将等待任何未落实的 ALTER TABLE 事务。这些事务将获取针对目录表中的相关行的互斥锁定，并且互斥锁定必须先终止，装入操作才能继续。这意味着装入操作运行期间，可能没有未落实的 ALTER TABLE ...ATTACH、DETACH 或 ADD PARTITION 事务。将拒绝目标为拆离或相连的数据分区的所有输入源记录，但可从异常表中检索它们（如果指定了异常表）。会有一条参考消息写入消息文件，以指示某些目标表数据分区处于相连或拆离状

态。针对对应于目标表的相关目录表行的锁定使得用户无法通过在 load 实用程序运行时发出 ALTER TABLE ...ATTACH、DETACH 或 ADD PARTITION 操作来更改目标表的分区。

处理无效行

当 load 实用程序遇到的记录不属于任何可视数据分区时，将拒绝该记录并且 load 实用程序继续进行处理。因为范围限制违例而拒绝的记录的个数不会显式显示出来，但会包括在拒绝的记录的总行数中。因为范围违例而拒绝记录不会增加行警告的数目。会有一条消息（SQL0327N）写入 load 实用程序消息文件，指示发现范围违例，但不会对每一个记录来记录消息。除了目标表中的所有列之外，异常表还包括用于描述特定行发生的类型违例的列。包含无效数据的行（包括未分区的数据）将写至转储文件。

因为异常表插入成本很高，所以可以控制插入到异常表中的约束违例。例如，load 实用程序的缺省行为是将本来有效但因为范围限制或唯一约束违例而拒绝的行插入到异常表中。通过对 FOR EXCEPTION 子句分别指定 NORANGEEXC 或 NOUNIQUEEXC 可以关闭此行为。如果指定不应将这些约束违例插入到异常表中，或者未指定异常表，那么有关违反范围限制或唯一约束的行的信息将会丢失。

历史记录文件

如果目标表已分区，那么相应的历史记录文件条目不会包括目标表跨越的表空间列表。另一操作详细程度标识（“R”而不是“T”）指示对分区表运行了装入操作。

终止装入操作

终止装入替换操作将完全截断所有可视数据分区，而终止装入插入操作会将所有可视数据分区截断至装入前的长度。如果联机装入操作在装入复制阶段失败，那么在终止该操作期间，索引会变得无效。在终止接触索引的脱机装入操作时，索引也会变得无效（因为重建索引方式或者增量维护期间插入了键而使得索引处于不一致状态，从而导致索引无效）。将数据装入到多个目标中不会影响装入恢复操作，但将无法从装入阶段期间获取的一致点重新启动装入操作。在此情况下，如果对目标表进行分区，那么将忽略 SAVECOUNT 装入选项。此行为与将数据装入到 MDC 目标表中的行为一致。

生成列

如果生成列在任何分区、维或分布键中，那么会忽略 `generatedoverride` 文件类型修饰符并且 load 实用程序会生成值，就像指定了 `generatedignore` 文件类型修饰符一样。在此情况下，装入错误的生成列值可能导致将记录放置在错误的物理位置上，例如，错误的数据分区、MDC 块或数据库分区。例如，一旦记录放在错误的数据分区上，设置完整性就必须将其移至另一物理位置，这不能在联机设置完整性操作期间完成。

数据可访问性

当前联机装入算法扩展至分区表。在 LOAD 命令上指定的联机装入（ALLOW READ ACCESS）允许阅读器并行访问整个表，包括装入和非装入数据分区。

数据分区状态

在某些情况下，成功装入后可视数据分区可能切换至“设置完整性暂挂”状态和/或“只读访问”表状态。如果该表存在装入操作不能保留的约束，那么数据分区可能会置于这些状态。这种约束可能包括检查约束和拆离的具体化查询表。失败的装入操作会导致所有可视数据分区处于“装入暂挂”表状态。

错误隔离

不支持在数据分区级别进行错误隔离。隔离错误意味着在运行时未出现错误的数据分

区上继续装入，而在运行时出现错误的数据分区上停止装入。可在不同数据库分区间隔离错误，但 load 实用程序不能在一个可视数据分区子集上落实事务，而回滚其余可视数据分区。

其他注意事项

- 如果有任何索引标记为无效，那么不支持递增。如果索引需要重建或拆离的从属项需要使用 SET INTEGRITY 语句进行验证，那么认为索引无效。
- 同时支持装入到分区表中，这些表使用按范围分区、按散列分布或按维算法组织的任何组合进行分区。
- 对于包括受装入影响的对象和表空间标识列表的日志记录，这些日志记录的大小（LOAD START 和 COMMIT (PENDING LIST)）可能增长得很快，并且因此而降低可供其他应用程序使用的活动日志空间量。
- 当表同时进行了分区和分布时，分区数据库装入可能不会影响所有数据库分区。只有输出数据库分区上的对象才会更改。
- 在装入操作期间，分区表的内存消耗会随表数的增加而增加。注意，总增加量不是线性的，因为仅总内存需求的一小部分与数据分区数成正比。

LBAC 保护的数据装入注意事项

要成功装入到带有受保护行的表中，必须具有 LBAC（基于标号的访问控制）凭证。对于当前与目标表相关联的安全策略，必须同时提供有效安全标号或者可转换为有效标号的安全标号。

如果没有有效的 LBAC 凭证，装入会失败并返回错误（SQLSTATE 42512）。如果输入数据未包含安全标号或该安全标号未使用其内部二进制格式，那么可使用若干文件类型修饰符以允许继续装入。

将数据装入到带有受保护行的表中时，目标表将有一个数据类型为 DB2SECURITYLABEL 的列。如果输入数据行未包含该列的值，那么该行会被拒绝，除非在 load 命令中指定了 `usedefaults` 文件类型修饰符，此时将使用您拥有的安全策略（用于保护表）中对应写访问权的安全标号。如果没有对应写访问权的安全标号，那么该行将被拒绝，并且处理会继续进至下一行。

如果要将数据装入到带有受保护行的表中，并且输入数据包括数据类型为 DB2SECURITYLABEL 列的值，那么将数据插入到该表中时遵循相同规则。如果用于保护要装入行的安全标号（数据文件的该行中的安全标号）就是您能够写入的安全标号，那么该安全标号将用于保护该行。（换言之，它将写入数据类型为 DB2SECURITYLABEL 的列。）如果您无法写入受该安全标号保护的行，那么产生的结果取决于保护源表的安全策略的创建方式:

- 如果创建该策略的 CREATE SECURITY POLICY 语句包含 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么将会拒绝行。
- 如果 CREATE SECURITY POLICY 语句未包含该选项或者包含 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么数据文件中对应该行的安全标号将被忽略，而您拥有的对应写访问权的安全标号将用于保护该行。在这种情况下，不会发出任何错误或警告。如果没有对应写访问权的安全标号，那么该行将被拒绝，并且处理会继续进至下一行。

定界符注意事项

将数据装入到数据类型为 DB2SECURITYLABEL 的列中时，缺省情况下会将数据文件

中的值视为组成该安全标号的内部表示的实际字节数。但是，某些原始数据可能包含换行符，它们可能会被 LOAD 命令错误地解释为行定界符。如果发生此问题，请使用 delprioritychar 文件类型修饰符以确保字符定界符优先于行定界符。使用 delprioritychar 时，包含在字符定界符中的任何记录或列定界符不会被识别为定界符。即使没有任何值包含换行符，使用 delprioritychar 文件类型修饰符也很安全，但这会稍微降低装入的速度。

如果要装入的数据为 ASC 格式，那么您可能必须执行一个额外步骤，以避免已装入的安全标号和安全标号名称中包含尾部空格。ASCII 格式将列位置用作定界符，所以装入到变长字段中时可能出现此情况。请使用 striptblanks 文件类型修饰符截断所有尾部空格。

非标准安全标号值

还可装入这样的数据文件，其安全标号值是包含安全标号中的组件的值的字符串，例如，*S:(ALPHA,BETA)*。为此，必须使用文件类型修饰符 seclabelchar。使用 seclabelchar 时，数据类型为 DB2SECURITYLABEL 的列的值将被视为字符串常量，包含对应安全标号的字符串格式的安全标号。如果字符串的格式不正确，那么不会插入该行，并且会返回警告（SQLSTATE 01H53）。如果该字符串未表示保护表的安全策略中的有效安全标号，那么不会插入该行，并且会返回警告（SQLSTATE 01H53）。

还可装入数据文件，其安全标号列的值是安全标号名称。要装入此类文件，必须使用文件类型修饰符 seclabelname。使用 seclabelname 时，数据类型为 DB2SECURITYLABEL 的所有列值将被视为包含现有安全标号名称的字符串常量。如果不存在具有所指示的用于保护表的安全策略名称的安全标号，那么不会装入行，并且会返回警告（SQLSTATE 01H53）。

被拒绝的行

在装入期间被拒绝的行将发送至转储文件或异常表（如果在 LOAD 命令中指定了该文件或表的话），这取决于拒绝行的原因。由于解析错误而被拒绝的行将发送至转储文件。违反安全策略的行将发送至异常表。

示例

对于所有示例，输入数据文件 myfile.del 将使用 DEL 格式。所有示例都将数据装入到名为 REPS 的表中，该表是使用以下语句创建的:

```
create table reps (row_label db2securitylabel,
id integer,
name char(30))
security policy data_access_policy
```

对于此示例，假定输入文件包含缺省格式的安全标号:

```
db2 load from myfile.del of del modified by delprioritychar insert into reps
```

对于此示例，假定输入文件包含安全标号字符串格式的安全标号:

```
db2 load from myfile.del of del modified by seclabelchar insert into reps
```

对于此示例，假定输入文件包含安全标号列的安全标号名称:

```
db2 load from myfile.del of del modified by seclabelname insert into reps
```

标识列装入注意事项

无论输入数据是否具有标识列值，都可以使用 load 实用程序将数据装入到包含标识列的表中。

如果未使用与标识相关的文件类型修饰符，那么该实用程序会遵循下列规则来工作:

- 如果标识列是 GENERATED ALWAYS 列，那么每当输入文件中的相应行缺少标识列值，或者显式指定了 NULL 值时，会为表行生成标识值。如果对标识列指定了非空值，那么会拒绝该行（SQL3550W）。
- 如果标识列是 GENERATED BY DEFAULT 列，load 实用程序就会使用用户提供的值（如果提供了这些值）；如果数据丢失或者显式指定了 NULL，就会生成值。

除了通常对标识列数据类型（即 SMALLINT、INT、BIGINT 或 DECIMAL）的值执行的验证操作以外，load 实用程序不会对用户提供的标识值执行任何其他验证操作。不报告重复值。

在大多数情况下，load 实用程序无法保证对各行指定标识列值的顺序与这些行在数据文件中的出现顺序相同。由于 load 实用程序以并行方式对标识列值的指定进行管理，所以这些值按任意顺序指定。它的例外情况如下所示:

- 在单一分区数据库中，当 CPU_PARALLELISM 设置为 1 时，不以并行方式处理行。在此情况下，将按照各行在数据文件参数中的出现顺序来隐式地指定标识列值。
- 在多分区数据库中，如果标识列位于分布键中以及存在单分区代理程序（即，未指定多个分区代理程序或 anyorder 文件类型修饰符），那么将按照各行在数据文件中的出现顺序来指定标识列值。

将表装入到分区数据库中时，如果该表在数据库的分区键中具有标识列并且未指定 identityoverride 修饰符，那么不能指定 SAVECOUNT 选项。当分区键中存在标识列并且正在生成标识值时，在至少一个数据库分区上从装入阶段重新启动装入操作需要从装入阶段开始时重新启动整个装入操作，这意味着不可能有任何一致点。

注: 如果符合下列所有条件，那么不允许执行 load RESTART 操作:

- 要装入的表位于分区数据库环境中，并且它包含至少一个标识列，该标识列位于分布键中或由分布键中的生成列引用。
- 未指定 identityoverride 修饰符。
- 失败的上一个装入操作包括装入在装入阶段后失败的数据库分区。

应改为发出 load TERMINATE 或 REPLACE 操作。

有三种相互排斥的方法可用来简化将数据装入到包含标识列的表中的操作: identitymissing、identityignore 和 identityoverride 文件类型修饰符。

在没有标识列的情况下装入数据

如果输入数据文件未包含任何标识列值（甚至未包含 NULL 值），那么 identitymissing 修饰符可以使您更方便地装入包含标识列的表。例如，考虑使用以下 SQL 语句定义的表:

```
create table table1 (c1 varchar(30),
                     c2 int generated by default as
                        identity,
                     c3 decimal(7,2),
                     c4 char(1))
```

如果想要将文件 load.del 中的数据装入到 TABLE1 中，该文件是从未包含标识列的表中导出的，请参阅以下示例:

```
Robert, 45.2, J
Mike, 76.9, K
Leo, 23.4, I
```

装入此文件的一种方法是通过 LOAD 命令显式列示要装入的列，如下所示:

```
db2 load from load.del of del replace into table1 (c1, c3, c4)
```

但是，对于包含许多列的表来说，此语法难以使用并且容易出错。另一种装入此文件的方法是使用 identitymissing 文件类型修饰符，如下所示:

```
db2 load from load.del of del modified by identitymissing
      replace into table1
```

此命令将导致数据文件中的三列装入到 TABLE1 的 c1、c3 和 c4 中。将为 c2 中的每列生成值。

在具有标识列的情况下装入数据

identityignore 修饰符向 load 实用程序指示: 即使输入数据文件包含标识列数据，也应该忽略该数据，并且应该为每一列生成标识值。例如，一位用户想将数据文件（load.del）中的数据装入到上面定义的 TABLE1 中，该数据文件包含以下数据:

```
Robert, 1, 45.2, J
Mike, 2, 76.9, K
Leo, 3, 23.4, I
```

如果用户提供的值 1、2 和 3 未用于标识列，那么可以发出以下 LOAD 命令:

```
db2 load from load.del of del method P(1, 3, 4)
      replace into table1 (c1, c3, c4)
```

同样，如果该表包含许多列，那么此方法可能难以使用并且容易出错。identityignore 修饰符可以将语法简化为:

```
db2 load from load.del of del modified by identityignore
      replace into table1
```

装入包含用户提供的值的数据

identityoverride 修饰符用来将用户提供的值装入到包含 GENERATED ALWAYS 标识列的表中。当从另一数据库系统迁移数据并且必须将表定义为 GENERATED ALWAYS 时，或者在将使用 ROLLFORWARD DATABASE 命令 DROPPED TABLE RECOVERY 选项恢复的数据装入到表中时，此修饰符非常有用。当使用了此修饰符时，将拒绝任何未包含标识列数据（或者包含 NULL 数据）的行（SQL3116W）。还应该注意，使用此修饰符时，可能会违反 GENERATED ALWAYS 列的唯一性属性。在这种情况下，执行 load TERMINATE 操作，然后接着执行 load INSERT 或 REPLACE 操作。

生成列装入注意事项

无论输入数据是否具有生成列值，都可以将数据装入到包含（非标识）生成列的表中。load 实用程序生成列值。

如果未使用任何与生成列相关的文件类型修饰符，load 实用程序就会依照下列规则工作:

- 当数据文件中相应的行缺少生成列的值或提供了 NULL 值时，将创建生成列值。如果为生成列提供了非空值，那么将拒绝该行（SQL3550W）。
- 如果为不可空生成列创建了 NULL 值，那么将拒绝整行数据（SQL0407N）。例如，如果将不可空生成列定义为两个表列之和，但这两个表列在数据文件中包含 NULL 值，就会发生这种情况。

有三种相互排斥的方法可用来简化将数据装入到包含生成列的表中的操作：generatedmissing、generatedignore 和 generatedoverride 文件类型修饰符。

在没有生成列的情况下装入数据

如果输入数据文件不包含表中的所有生成列的任何值（甚至不包含 NULL 值），那么 generatedmissing 修饰符能使您更方便地装入包含生成列的表。例如，考虑使用以下 SQL 语句定义的表：

```
CREATE TABLE table1 (c1 INT,
                     c2 INT,
                     g1 INT GENERATED ALWAYS AS (c1 + c2),
                     g2 INT GENERATED ALWAYS AS (2 * c1),
                     c3 CHAR(1))
```

如果想要将文件 load.del 中的数据装入到 TABLE1 中，该文件是从未包含任何生成列的表中导出的，请参阅以下示例：

```
1, 5, J
2, 6, K
3, 7, I
```

装入此文件的一种方法是通过 LOAD 命令显式列示所要装入的列，如下所示：

```
DB2 LOAD FROM load.del of del REPLACE INTO table1 (c1, c2, c3)
```

但是，对于包含许多列的表来说，此语法难以使用并且容易出错。另一种装入此文件的方法是使用 generatedmissing 文件类型修饰符，如下所示：

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedmissing
      REPLACE INTO table1
```

此命令将导致数据文件的三列装入到 TABLE1 的 c1、c2 和 c3 中。由于 generatedmissing 修饰符，将自动生成 TABLE1 的 g1 列和 g2 列的值，并且这些值不会映射至任何数据文件列。

在具有生成列的情况下装入数据

generatedignore 修饰符向 load 实用程序指示：即使输入数据文件包含目标表中所有生成列的数据，也应该忽略该数据，并且应将计算值装入到每个生成列中。例如，如果想要将包含以下数据的数据文件（load.del）中的数据装入到上面定义的 TABLE1 中：

```
1, 5, 10, 15, J
2, 6, 11, 16, K
3, 7, 12, 17, I
```

如果未使用与生成列相关的文件类型修饰符，那么用户提供的非空值 10、11 和 12（用于 g1）以及 15、16 和 17（用于 g2）将导致拒绝该行（SQL3550W）。为了避免这种情况，用户可以发出以下 LOAD 命令：

```
DB2 LOAD FROM load.del of del method P(1, 2, 5)
      REPLACE INTO table1 (c1, c2, c3)
```

同样，如果该表包含许多列，那么此方法可能难以使用并且容易出错。generatedignore 修饰符可以将语法简化为:

```
DB2 LOAD FROM load.del of del MODIFIED BY generatedignore
      REPLACE INTO table1
```

此命令将导致数据文件中的列装入到 TABLE1 的 c1（数据为 1、2 和 3）、c2（数据为 5、6、7）和 c3（数据为 J、K 和 I）中。由于 generatedignore 修饰符，将自动生成 TABLE1 的 g1 列和 g2 列的值，并且将忽略数据文件列（10、11 和 12 以及 15、16 和 17）。

装入包含用户提供的值的数据

generatedoverride 修饰符用来将用户提供的值装入到包含生成列的表中。当从另一个数据库系统迁移数据，或者在将使用 ROLLFORWARD DATABASE 命令 RECOVER DROPPED TABLE 选项恢复的数据装入到表中时，此修饰符非常有用。当使用了此修饰符时，将拒绝任何未包含不可空生成列数据（或者包含 NULL 数据）的行（SQL3116W）。

使用此修饰符时，装入操作完成后将使表处于“设置完整性暂挂”状态。要使该表脱离“设置完整性暂挂”状态，而不验证用户提供的值，请发出以下命令:

```
SET INTEGRITY FOR table-name GENERATED COLUMN IMMEDIATE
    UNCHECKED
```

要使该表脱离“设置完整性暂挂”状态并强制验证用户提供的值，请发出以下命令:

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED
```

如果生成列在任何分区、维或分布键中，那么会忽略 generatedoverride 修饰符并且 load 实用程序会生成值，就像指定了 generatedignore 修饰符一样。这样做是为了避免用户提供的生成列值与它的生成列定义相冲突，在这种情况下，会将生成的记录放置在错误的物理位置，例如，错误的数据分区、MDC 块或数据库分区。

注: 在以下情况下，装入不支持生成列值: 其中一个生成列表达式包含受防护的（FENCED）用户定义的函数。如果尝试装入到这样的表中，装入操作会失败。但是，通过使用 generatedoverride 文件类型修饰符，可以为这些类型的生成列提供您自己的值。

将版本 7 或更旧版本的客户机与版本 8 或更高版本的服务器配合使用的注意事项

如果在版本 7 或更旧版本的客户机与版本 8 或更高版本的服务器之间启动装入操作，那么 load 实用程序将使包含生成列的表置于“设置完整性暂挂”状态。如果由于使用了版本 7 或更旧版本的客户机将数据装入到包含生成列的表中，从而使表处于“设置完整性暂挂”状态，那么发出以下语句脱离该状态并强制生成值:

```
SET INTEGRITY FOR table-name IMMEDIATE CHECKED FORCE GENERATED;
```

使用 CURSOR 文件类型来移动数据

通过在使用 LOAD 命令时指定 CURSOR 文件类型，可以将 SQL 查询结果直接装入到目标表中，而不必创建中间导出的文件。

此外，通过在 SQL 查询中引用昵称，在 DECLARE CURSOR 语句中使用 DATABASE 选项，或者在使用 API 接口时使用 sqlu_remotefetch_entry 介质条目，可从另一个数据库装入数据。

有三种方法可用于通过使用 CURSOR 文件类型来移动数据。第一种方法使用命令行处理器（CLP），第二种方法使用 API，第三种方法使用 ADMIN_CMD 过程。下表概述了 CLP 与 ADMIN_CMD 过程之间的主要差别。

表 26. . CLP 与 ADMIN_CMD 过程之间的差别。

差别	CLP	ADMIN_CMD_procedure
语法	游标所使用的查询语句和源数据库是使用 DECLARE CURSOR 语句在 LOAD 命令之外定义的。	游标所使用的查询语句和源数据库是使用 LOAD From（DATABASE 数据库别名查询语句）在 LOAD 命令内定义的。
用于访问另一个数据库的用户权限	如果数据所在的数据库不是当前连接至的数据库，那么必须在 DECLARE CURSOR 语句中使用 DATABASE 关键字。还可以在同一个语句中指定用户标识和密码。如果未在 DECLARE CURSOR 语句中指定用户标识和密码，那么使用为源数据库连接显式指定的用户标识和密码来访问目标数据库。	如果数据所在的数据库不是当前连接至的数据库，那么在查询语句之前必须在 LOAD 命令中使用 DATABASE 关键字。需要使用为源数据库连接显式指定的用户标识和密码来访问目标数据库。不能为源数据库指定用户标识或密码。因此，如果在建立与目标数据库的连接时未指定用户标识和密码，或者指定的用户标识和密码不能用于认证源数据库，那么不能使用 ADMIN_CMD 过程来执行装入。

要从 CLP 中执行 LOAD FROM CURSOR 操作，首先必须对 SQL 查询声明游标。一旦声明了游标，就可以发出 LOAD 命令，并将所声明游标的名称用作 *cursorname*，将 CURSOR 用作文件类型。

例如:

1. 假定源表和目标表位于同一数据库中，并且它们的定义如下所示:

   表 ABC.TABLE1 有 3 列:

   - ONE INT
   - TWO CHAR(10)
   - THREE DATE

   表 ABC.TABLE2 有 3 列:

   - ONE VARCHAR
   - TWO INT
   - THREE DATE

   执行下列 CLP 命令会将 ABC.TABLE1 中的所有数据装入到 ABC.TABLE2 中:

   ```
   DECLARE mycurs CURSOR FOR SELECT TWO, ONE, THREE FROM abc.table1
         LOAD FROM mycurs OF cursor INSERT INTO abc.table2
   ```

注: 以上示例显示了如何通过 CLP 来从 SQL 查询装入。但是，还可以通过 db2Load API 从 SQL 查询装入。定义 sqlu_media_list 结构的 *piSourceList* 以使用 *sqlu_statement_entry* 结构和 SQLU_SQL_STMT 介质类型，并将 *piFileType* 值定义为 SQL_CURSOR。

2. 假定源表与目标表位于不同的数据库中，并且它们的定义如下所示:

数据库“dbsource”中的表 ABC.TABLE1 有 3 列:

- ONE INT
- TWO CHAR(10)
- THREE DATE

数据库“dbtarget”中的表 ABC.TABLE2 有 3 列:

- ONE VARCHAR
- TWO INT
- THREE DATE

只要您启用了联合功能并且编目了数据源（“dsdbsource”），就可以对源数据库声明昵称，然后对此昵称声明游标，并调用带有 FROM CURSOR 选项的 LOAD 命令，如以下示例所示:

```
CREATE NICKNAME myschema1.table1 FOR dsdbsource.abc.table1
DECLARE mycurs CURSOR FOR SELECT TWO,ONE,THREE FROM myschema1.table1
LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

也可以使用 DECLARE CURSOR 语句的 DATABASE 选项，如以下示例所示:

```
DECLARE mycurs CURSOR DATABASE dbsource USER dsciaraf USING mypasswd
FOR SELECT TWO,ONE,THREE FROM abc.table1
   LOAD FROM mycurs OF cursor INSERT INTO abc.table2
```

使用 DECLARE CURSOR 语句的 DATABASE 选项（在使用装入 API 时又称为远程访存介质类型）与昵称方法相比具有一些优点:

性能

对使用远程访存介质类型的数据的访存紧密集成在装入操作中。与昵称方法相比，访存记录时转换的层数更少。此外，当源表与目标表完全相同地分布在多分区数据库中时，load 实用程序可让数据访存并行进行，这样可以进一步提高性能。

易于使用

不需要启用联合、定义远程数据源或声明昵称。只需要指定 DATABASE 选项（必要时还需要指定 USER 和 USING 选项）。

虽然此方法可与已编目数据库配合使用，但使用昵称为访存各种数据源（不能仅仅是编目）提供了强大的功能。

为了支持此远程访存功能，load 实用程序将使用支持 SOURCEUSEREXIT 工具的基础结构。load 实用程序衍生了这样一个进程，该进程作为管理与源数据库的连接并执行访存的应用程序执行。此应用程序与自己的事务相关联，但不与运行 load 实用程序的事务相关联。

注:

1. 上一个示例显示了如何使用 DECLARE CURSOR 语句的 DATABASE 选项来通过 CLP 从针对已编目数据库的 SQL 查询装入。但是，通过将 *db2LoadStruct* 结构的 *piSourceList* 和 *piFileTypevalues* 分别定义为使用 sqlu_remotefetch_entry 介质条目和 SQLU_REMOTEFETCH 介质条目，还可通过 db2Load API 来从针对已编目数据库的 SQL 查询装入。
2. 如上一个示例所示，SQL 查询的源列类型不需要与它们的目标列类型完全相同，尽管它们必须兼容。

限制

从使用 DATABASE 选项定义的游标装入（相当于将 sqlu_remotefetch_entry 介质条目与 db2Load API 配合使用）时，下列限制适用:

1. 不能同时指定 SOURCEUSEREXIT 选项。
2. 不支持 METHOD N 选项。
3. 不支持 `usedefaults` 文件类型修饰符。

传播从属立即登台表

如果要装入的表是带有立即传播属性的登台表的底层表，并且以插入方式进行装入操作，那么将以递增方式后续传播至从属立即登台表。

在递增传播期间，与底层表中的追加行对应的行将追加至登台表。如果底层表很大而追加数据的量很少，那么递增传播会比较快。如果使用登台表来刷新其从属延迟具体化查询表，那么也可以改进性能。也存在不允许递增传播的情况，此时登台表将被标记为不完整。即，CONST_CHECKED 列的登台字节将具有值 F。在此状态下，不能使用登台表来刷新其从属延迟具体化查询表，并且具体化查询表维护过程需要完全刷新。

如果表处于不完整状态并且指定了 INCREMENTAL 选项，但不能对表进行递增传播，那么会返回错误。如果发生下列任一情况，系统将关闭直接数据传播并将表状态设置为不完整:

- 在上次对登台表的底层表执行完整性检查之后，已在该底层表中执行装入替换操作，或者已激活 NOT LOGGED INITIALLY WITH EMPTY TABLE 选项。
- 登台表的从属具体化查询表或者登台表已使用 REPLACE 或 INSERT 方式装入。
- 在完整性检查期间使用 FULL ACCESS 选项传播登台表之前，底层表已脱离“设置完整性暂挂”状态。
- 已经以非递增方式检查了登台表的底层表的完整性。
- 包含登台表或其底层表的表空间已前滚至某个时间点，并且该登台表及其底层表位于不同的表空间中。

如果登台表在 SYSCAT.TABLES 目录的 CONST_CHECKED 列中具有 `W` 值，并且未指定 NOT INCREMENTAL 选项，那么将以递增方式传播至登台表，并且 SYSCAT.TABLES 的 CONST_CHECKED 列将标记为 `U` 以指示系统并未验证所有数据。

以下示例说明在登台表 G1 及其从属延迟具体化查询表 AST1 的底层表 UT1 中进行的装入插入操作。在此方案中，将以递增方式检查 UT1 的完整性以及刷新 AST1:

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1,G1 IMMEDIATE CHECKED;

REFRESH TABLE AST1 INCREMENTAL;
```

刷新从属立即具体化查询表

如果使用 INSERT 选项装入立即刷新具体化查询表的底层表，并对使用 REFRESH IMMEDIATE 定义的从属具体化查询表执行 SET INTEGRITY 语句，那么会导致递增刷新具体化查询表。

在递增刷新期间，与底层表中的追加行对应的行将更新并插入至具体化查询表。如果底层表很大而追加数据的量很少，那么递增刷新的速度会比较快。也存在不允许递增刷新的情况，此时将使用完全刷新（即，再计算具体化查询表定义查询）。

当指定了 INCREMENTAL 选项，但不能进行具体化查询表递增处理时，在下列情况下，将返回错误:

- 在上次对具体化查询表的底层表执行完整性检查之后，已在该底层表中执行装入替换操作，或者已激活 NOT LOGGED INITIALLY WITH EMPTY TABLE 选项。
- 已使用 REPLACE 或 INSERT 方式装入具体化查询表。
- 在完整性检查期间使用 FULL ACCESS 选项刷新具体化查询表之前，底层表已脱离“设置完整性暂挂”状态。
- 已经以非递增方式检查了具体化查询表的底层表的完整性。
- 具体化查询表在迁移前处于设置完整性暂挂状态。
- 包含具体化查询表或其底层表的表空间已前滚至某个时间点，并且该具体化查询表及其底层表位于不同的表空间中。

如果具体化查询表在 SYSCAT.TABLES 目录的 CONST_CHECKED 列中有一个或多个 W 值，并且如果未在 SET INTEGRITY 语句中指定 NOT INCREMENTAL 选项，那么该表将进行递增刷新，并且 SYSCAT.TABLES 的 CONST_CHECKED 列将标记为 U 以指示系统并未验证所有数据。

以下示例说明在具体化查询表 AST1 的底层表 UTI 中进行的装入插入操作。将检查 UT1 的数据完整性，并且会将 UT1 置于无数据移动方式。一旦 AST1 的递增刷新完成，UT1 就将回复完全访问状态。在此方案中，将以递增方式检查 UT1 的完整性以及刷新 AST1。

```
LOAD FROM IMTFILE1.IXF of IXF INSERT INTO UT1;
LOAD FROM IMTFILE2.IXF of IXF INSERT INTO UT1;
SET INTEGRITY FOR UT1 IMMEDIATE CHECKED;
REFRESH TABLE AST1;
```

的多维集群注意事项

在将数据装入多维集群（MDC）表时，下列限制适用:

- 不支持 LOAD 命令的 SAVECOUNT 选项。
- 由于这些表管理它们自己的可用空间，所以不支持 totalfreespace 文件类型修饰符。
- MDC 表需要 anyorder 文件类型修饰符。如果对 MDC 表执行装入，但未指定 anyorder 修饰符，那么实用程序将显式启用该修饰符。

对 MDC 表使用 LOAD 命令时，将按以下方式处理唯一约束违例:

- 如果执行装入操作前该表包含唯一键，并且将重复记录装入该表，那么将保留原始记录，并且在删除阶段删除新记录。
- 如果执行装入操作前该表未包含唯一键，并且将唯一键和重复记录都装入该表，那么将只装入其中一个带有唯一键的记录，并且在删除阶段删除其他记录。

注: 没有确切的技术可用来确定将要装入的记录以及将要删除的记录。

性能注意事项

为了提高 load 实用程序在装入 MDC 表时的性能，应该增大 *util_heap_sz* 数据库配置参数值。当有更多内存可供该实用程序使用时，MDC 装入算法的性能会显著提高。这将减少在装入阶段执行数据集群时的磁盘 I/O 次数。当指定了 LOAD 命令的 DATA BUFFER 选项时，还应增大它的值。如果使用 LOAD 命令来并发地装入若干个 MDC 表，那么应相应地增大 *util_heap_sz*。

由于所有 MDC 表都有块索引，所以 MDC 装入操作始终包括构建阶段。

在装入阶段，将执行附加的记录以维护块映射。对于分配的每个扩展数据块，大约有两个附加的日志记录。为了确保性能良好，在设置 *logbufsz* 数据库配置参数地值时应该考虑此情况。

在将数据装入 MDC 表时，将使用一个带有索引的系统临时表。该表的大小与装入的单个单元数成正比。该表中每一行的大小与 MDC 维键的大小成正比。为了最大程度地减少装入操作期间处理此表时执行的磁盘 I/O 次数，请确保临时表空间的缓冲池足够大。

使用定制应用程序（用户出口）移动数据

装入 SOURCEUSEREXIT 选项提供了一种工具，load 实用程序可通过该工具执行定制脚本或可执行文件（此处称为*用户出口*）。

用户出口使用 load 实用程序正在读取的数据填充一个或多个命名管道。在多分区数据库中，可并行调用用户出口的多个实例以实现输入数据的并行性。

如第 136 页的图 5 所示，load 实用程序创建一个或多个命名管道并衍生一个进程以执行定制的可执行文件。在 load 实用程序读取数据的同时，用户出口将数据填充到命名管道中。

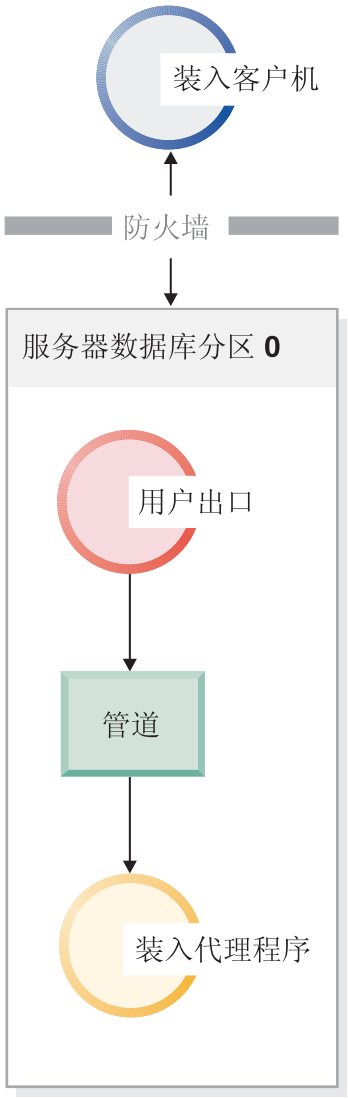


图 5. load 实用程序从管道读取传入的数据并对其进行处理。

填充到管道中的数据必须反映指定的装入选项，包括文件类型和任何文件类型修饰符。load 实用程序不会直接读取指定的数据文件。在执行用户出口时，指定的数据文件将作为自变量传递至用户出口。

调用用户出口

用户出口必须在 DB2 安装目录的类别子目录中（通常又称为 sqllib）。load 实用程序使用下列命令行参数来调用用户出口可执行文件:

```
<base pipename> <number of source media>
<source media 1> <source media 2> ... <user exit ID>
<number of user exits> <database partition number>
```

其中:

< base pipename >

是 load 实用程序创建的命名管道的基本名称，load 实用程序将从这个命名管道读取数据。load 实用程序为提供给 LOAD 命令的每个源文件创建一个管道，每个管道都追加 .xxx 作为结尾，其中 xxx 是提供的源文件的索引。例如，如果

向 LOAD 命令提供了 2 个源文件，并且传递至用户出口的 <base pipename> 自变量为 pipe123，那么用户出口应为其填充数据的两个命名管道为 pipe123.000 和 pipe123.001。在分区数据库环境中，load 实用程序将数据库分区（DBPARTITION）号 .yyy 追加至基本管道名，从而生成管道名 pipe123.xxx.yyy。

<number of source media>
是跟在后面的介质参数数目。

<source media 1> <source media 2> ...
是 LOAD 命令中指定的一个或多个源文件的列表。每个源文件都必须用双引号括起来。

<user exit ID>
是一个特殊值，当启用了 PARALLELIZE 选项时，此值非常有用。此整数值（从 1 到 N，其中 N 是衍生的用户出口总数）标识正在运行的用户出口的特定实例。未启用 PARALLELIZE 选项时，此值缺省为 1。

<number of user exits>
是一个特殊值，当启用了 PARALLELIZE 选项时，此值非常有用。此值表示当前正在运行的用户出口总数。未启用 PARALLELIZE 选项时，此值缺省为 1。

<database partition number>
是一个特殊值，当启用了 PARALLELIZE 选项时，此值非常有用。这是执行用户出口的数据库分区（DBPARTITION）号。未启用 PARALLELIZE 选项时，此值缺省为 0。

附加选项和功能

下节描述其他 SOURCEUSEREXIT 工具选项:

REDIRECT
此选项允许您将数据传递至 STDIN 句柄，或者从用户出口进程的 STDOUT 和 STDERR 句柄捕获数据。

INPUT FROM BUFFER < buffer >
允许将信息直接传递至用户出口的 STDIN 输入流。在衍生执行用户出口的进程后，load 实用程序将获取这个新进程的 STDIN 文件描述符并通过提供的缓冲区传递。该用户出口读取 STDIN 以获取该信息。load 实用程序仅使用 STDIN 将 <buffer> 的内容发送至用户出口，并不会解释或修改其内容。例如，如果将用户出口设计成从 STDIN 读取两个值（一个八字节用户标识和一个八字节密码），那么使用 C 编写的用户出口可执行文件可能包含下列代码行:

```
rc = read (stdin, pUserID, 8);
rc = read (stdin, pPasswd, 8);
```

用户可以使用 INPUT FROM BUFFER 选项来传递此信息，如以下 LOAD 命令所示:

```
LOAD FROM myfile1 OF DEL INSERT INTO table1
SOURCEUSEREXIT myuserexit1 REDIRECT INPUT FROM BUFFER myuseridmypasswd
```

注: load 实用程序将 <buffer> 的大小限制为 LOB 值的最大大小。但在命令行处理器（CLP）中，<buffer> 的大小被限制为 CLP 语句的最大大小。在 CLP 中，还建议 <buffer> 仅包含传统 ASCII 字符。如果使用 db2Load API 调用 load 实用程序，或者使用 INPUT FROM FILE 选项，就可以避免这些问题。

INPUT FROM FILE < filename>

允许将客户端文件的内容直接传递至用户出口的 STDIN 输入流。此选项与 INPUT FROM BUFFER 选项几乎完全相同，但此选项可避免潜在 CLP 限制。文件名必须是标准客户端文件，并且大小不能超过 LOB 值的最大大小。

OUTPUT TO FILE < filename>

允许将用户出口进程中的 STDOUT 和 STDERR 流捕获到服务器端文件中。在衍生执行用户出口可执行文件的进程后，load 实用程序将 STDOUT 和 STDERR 句柄从这个新进程重定向至指定的文件名。此选项在调试和记录用户出口中的错误和活动时非常有用。文件名必须是标准服务器端文件。启用 PARALLELIZE 选项后，每个用户出口都有一个文件，并且每个文件追加了三位数字标识，如 *filename.000*。

PARALLELIZE

此选项可通过同时调用多个用户出口进程来提高进入 load 实用程序的数据吞吐量。此选项仅适用于多分区数据库。如果在装入操作期间将数据分布在多个数据库分区上，那么调用的用户出口实例数等于分布代理程序数，否则它将等于装入代理程序数。

传递至每个用户出口的 <user exit ID>、<number of user exits> 和 <database partition number> 自变量分别反映唯一标识（从 1 到 N）、用户出口总数（N）以及运行用户出口实例的数据库分区（DBPARTITION）号。您应该确保每个用户出口进程写至命名管道的任何数据都不与其他并行进程重复。虽然用户出口应用程序可以有许多方法来完成这项任务，但这些值对确保数据不重复很有帮助。例如，如果每个数据记录包含唯一整数列值，那么用户出口应用程序可使用 <user exit ID> 和 <number of user exits> 值以确保每个用户出口实例将唯一结果集返回至其命名管道。用户出口应用程序可能按如下方式使用 **MODULUS** 属性:

```
i = <user exit ID>
N = <number of user exits>

foreach record
{
      if ((unique-integer MOD N) == i)
   {
         write this record to my named-pipe
   }
}
```

衍生的用户出口进程数取决于对数据库分区指定的分布方式:

1. 如第 139 页的图 6 所示，当指定未带 PARALLEL 的 PARTITION_AND_LOAD（缺省值）或 PARTITION_ONLY 时，将对每个预分布代理程序衍生一个用户出口进程。.

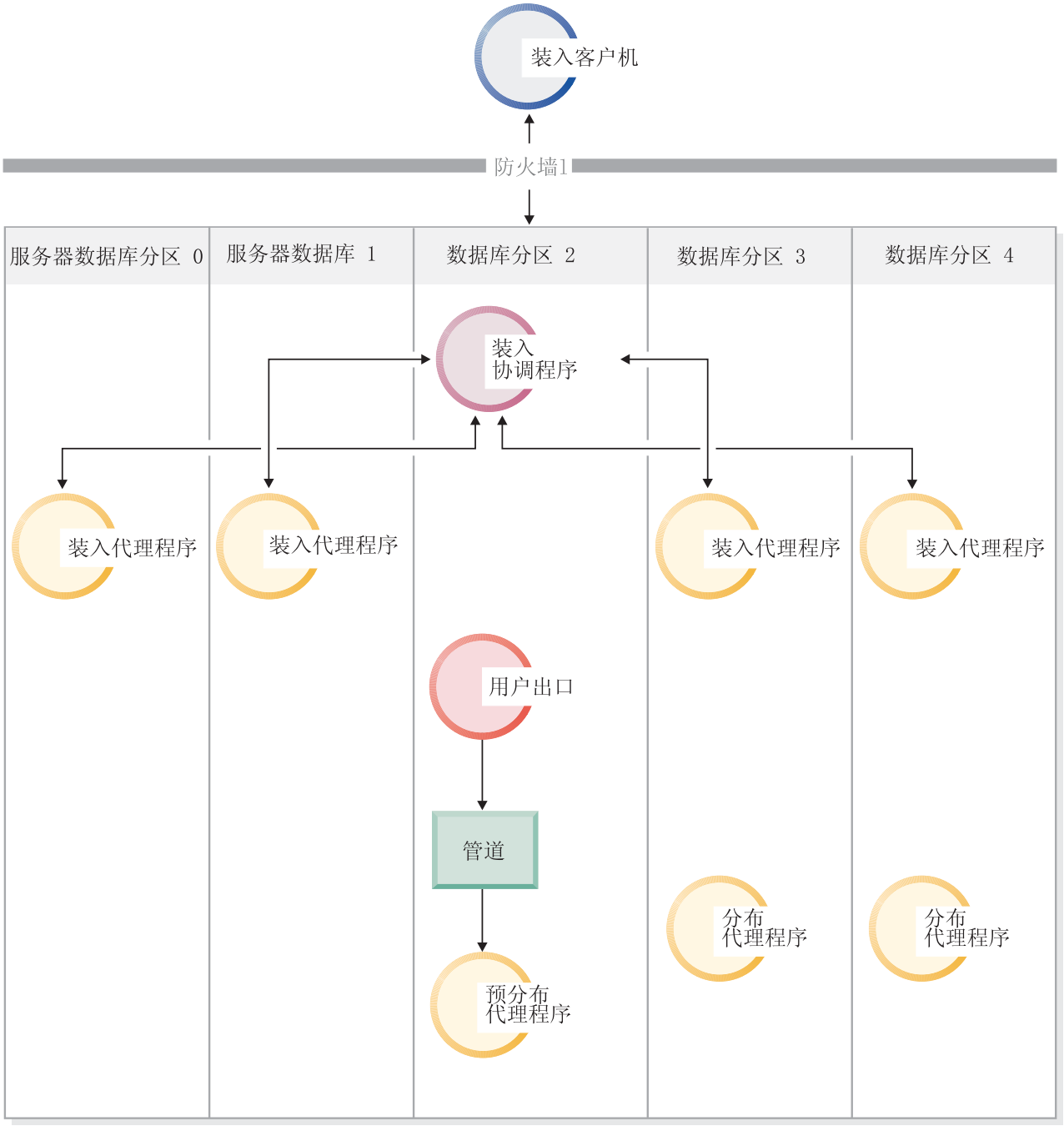


图 6. 指定未带 PARALLEL 的 PARTITION_AND_LOAD（缺省值）或 PARTITION_ONLY 时执行的各种任务。

2. 如第 140 页的图 7 所示，当指定带有 PARALLEL 的 PARTITION_AND_LOAD（缺省值）或 PARTITION_ONLY 时，将对每个分布代理程序衍生一个用户出口进程。

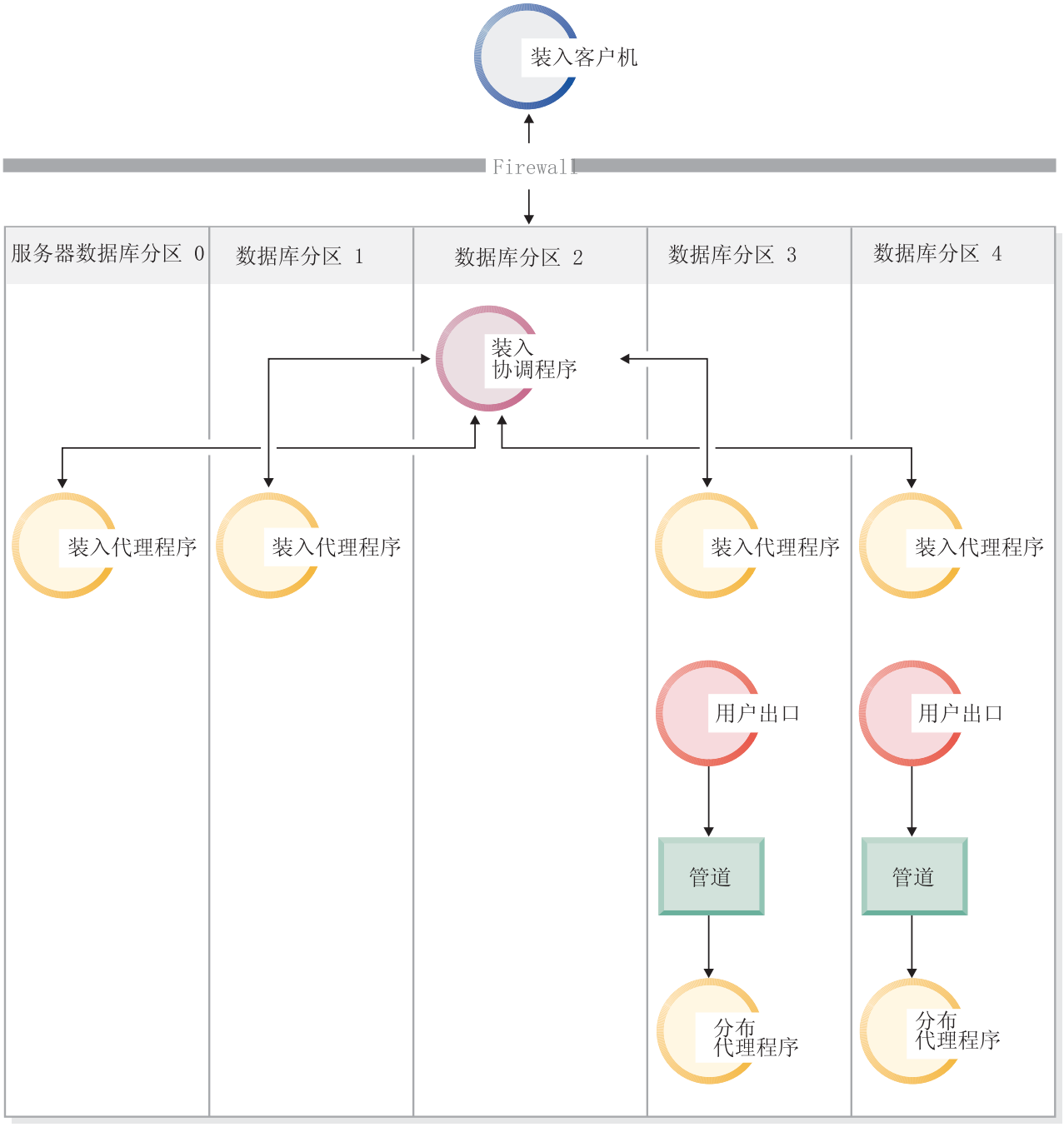


图 7. 指定带有 PARALLEL 的 PARTITION_AND_LOAD（缺省值）或 PARTITION_ONLY 时执行的各种任务。

3. 如第 141 页的图 8 所示，当指定 LOAD_ONLY 或 LOAD_ONLY_VERIFY_PART 时，将对每个装入代理程序衍生一个用户出口进程。

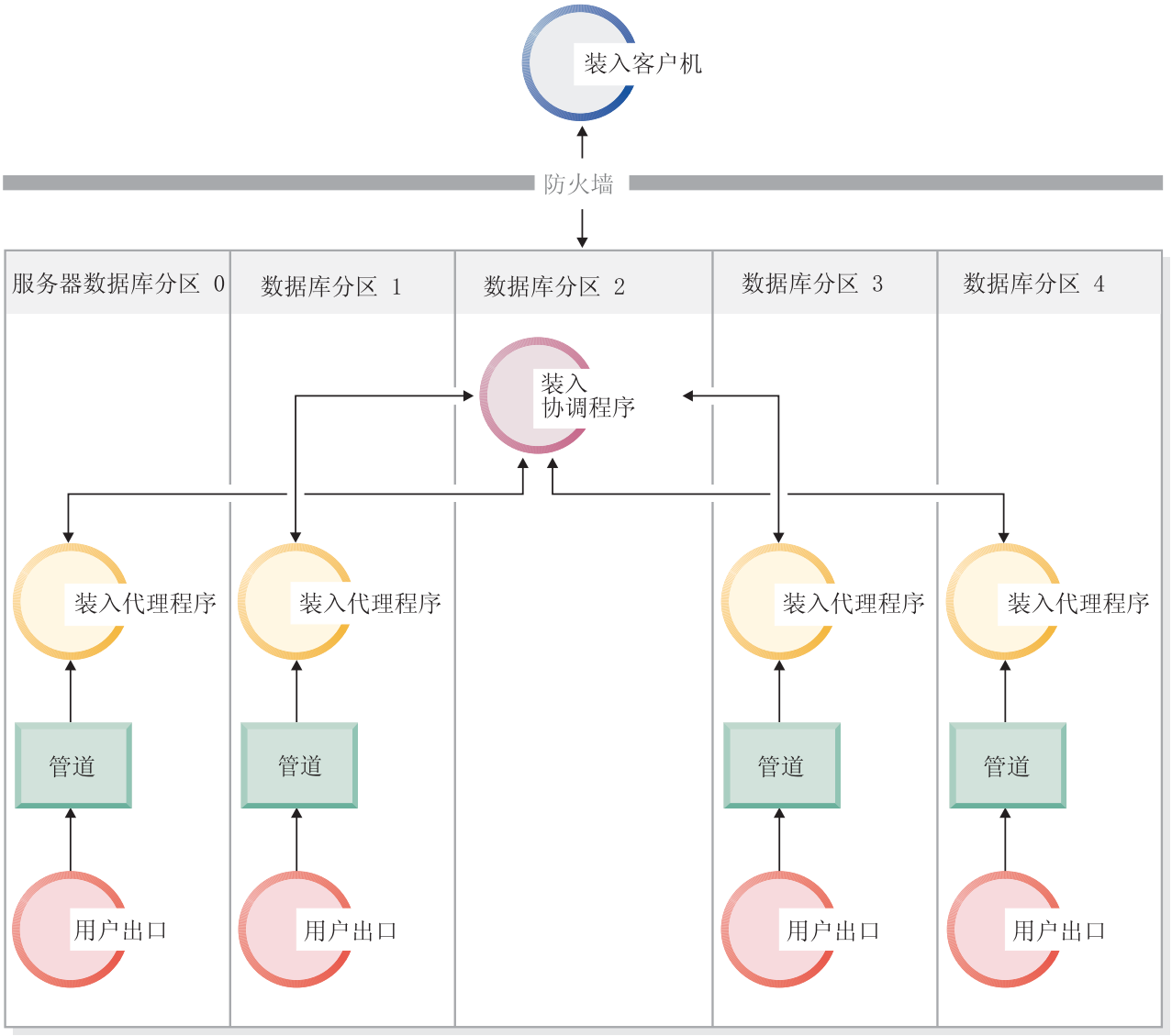


图 8. 指定 LOAD_ONLY 或 LOAD_ONLY_VERIFY_PART 时执行的各种任务。

限制

- 如果未指定 SOURCEUSEREXIT PARALLELIZE 选项，那么不支持 LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART partitioned-db-cfg 方式选项。

其他装入注意事项

并行性和装入

load 实用程序利用使用多个处理器或多个存储设备的硬件配置，如对称多处理器（SMP）环境。

通过使用 load 实用程序，有多种方法可用来并行处理大量数据。一种方法是通过使用多个存储设备，这允许在装入操作期间利用 I/O 并行性（请参阅第 142 页的图 9）。另一种方法涉及在 SMP 环境中使用多个处理器，这允许利用分区内并行性（请参阅第 142 页的

页的图 10）。两种方法可一起使用以提高数据装入速度。

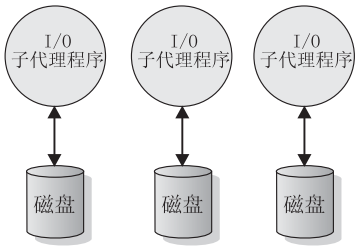


图 9. 在装入数据时利用 I/O 并行性

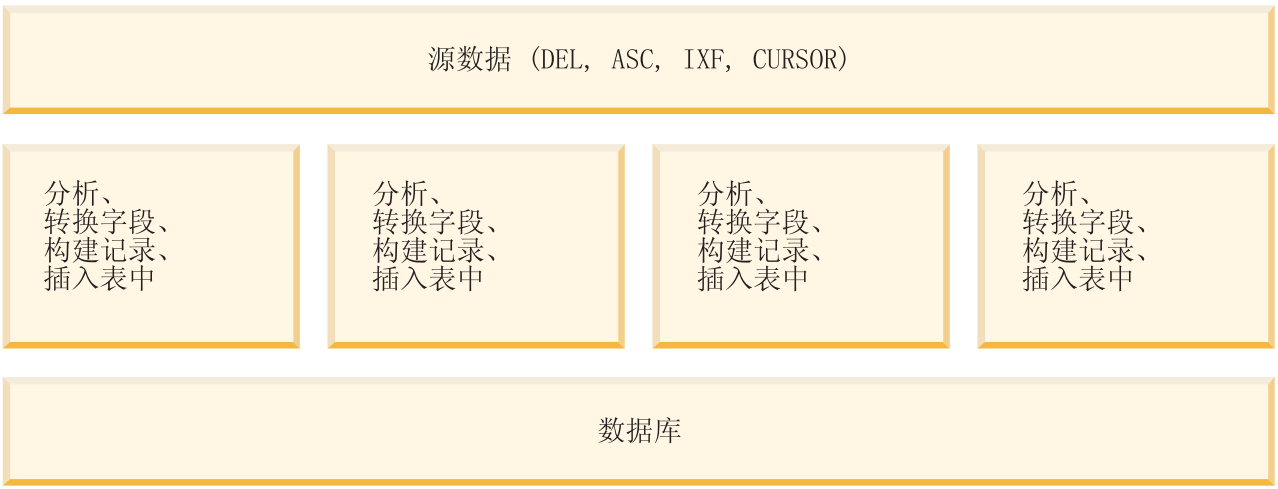


图 10. 在装入数据时利用分区内并行性

在装入操作期间创建索引

将在装入操作的构建阶段构建索引。可在 LOAD 命令中指定四种建立索引方式:

1. REBUILD。将重建所有索引。
2. INCREMENTAL。使用新数据扩展索引。
3. AUTOSELECT。load 实用程序自动决定是使用 REBUILD 还是 INCREMENTAL 方式。AUTOSELECT 是缺省值。如果正在执行装入替换操作，那么将使用 REBUILD 索引建立方式。否则，根据表中的现有数据量与新近装入的数据量的比率来选择建立索引方式。如果比率非常大，那么选择 INCREMENTAL 建立索引方式。否则，选择 REBUILD 建立索引方式。
4. DEFERRED。如果指定此方式，那么 load 实用程序不会尝试创建索引。索引将标记为需要刷新，并且可能会在第一次访问索引时强制重建。在下列任何一种情况下，不允许使用 DEFERRED 选项:
   - 指定了 ALLOW READ ACCESS 选项（它不会维护索引并且索引扫描程序需要有效索引）
   - 针对表定义了任何唯一索引
   - 正在装入 XML 数据（XML 路径索引是唯一的，并且缺省情况下，只要在表中添加 XML 列，就会创建该索引）

在取决于所选建立索引方式的类型的空间使用情况和记录方面，指定 ALLOW READ ACCESS 选项的装入操作需要特别注意。指定了 ALLOW READ ACCESS 选项时，load 实用程序将使索引可用于查询，即使正在重建索引也是如此。

当处于 ALLOW READ ACCESS 方式的装入操作指定 INDEXING MODE INCREMENTAL 选项时，load 实用程序将写入某些日志记录以保护索引树的完整性。写入的日志记录数目是插入键的数目的一部分，并且此数目比类似 SQL 插入操作需要的数目要少得多。处于 ALLOW NO ACCESS 方式并且指定了 INDEXING MODE INCREMENTAL 选项的装入操作仅写入正常空间分配日志之外的小型日志记录。

注: 只有在未指定 COPY YES 并且 *logindexrebuild* 配置参数设置为 ON 时，才会出现上述情况。

当处于 ALLOW READ ACCESS 方式的装入操作指定 INDEXING MODE REBUILD 选项时，新索引将在原始索引所在的表空间或系统临时表空间中构建为影子索引。在装入操作期间，原始索引保持原样并且可用，只有在装入操作结束并且以独占方式锁定该表时，新索引才会替换原始索引。如果装入操作失败并且回滚事务，那么原始索引保持原样。

缺省情况下，将在原始索引所在的表空间中构建影子索引。因为原始索引和新索引都同时保留下来，所以必须有足够的表空间才能同时容纳两个索引。如果装入操作异常中止，那么将释放用于构建新索引的其他空间。如果装入操作落实，那么将释放用于原始索引的空间，并且新索引成为当前索引。在原始索引所在的表空间中构建新索引时，几乎会同时替换原始索引。

如果在 SMS 表空间中构建索引，那么可以在带有 `.IN1` 后缀和 `.INX` 后缀的表空间目录中看到索引文件。这些后缀未指示哪一个是原始索引，哪一个是影子索引。但是，如果索引是在 DMS 表空间中构建的，那么您将看不到新的影子索引。

提高索引创建性能

在系统临时表空间中构建新索引

可在系统临时表空间中构建新索引以避免用完原始表空间中的空间。在使用 INDEXING MODE REBUILD 和 ALLOW READ ACCESS 选项时，USE <tablespace-name> 选项允许在系统临时表空间中重建索引。系统临时表空间可以是 SMS 或 DMS 表空间，但系统临时表空间的页大小必须与原始索引表空间的页大小相匹配。

如果装入操作未处于 ALLOW READ ACCESS 方式，或者建立索引方式不兼容，那么会忽略 USE <tablespace-name> 选项。只有 INDEXING MODE REBUILD 或 INDEXING MODE AUTOSELECT 选项才支持 USE <tablespace-name> 选项。如果指定了 INDEXING MODE AUTOSELECT 选项并且 load 实用程序选择以递增方式维护索引，那么会忽略 USE <tablespace-name> 选项。

即使原始装入操作未使用备用表空间，装入重新启动操作也可以使用备用表空间来构建索引。如果原始装入操作不是以 ALLOW READ ACCESS 方式发出的，那么不能以 ALLOW READ ACCESS 方式发出装入重新启动操作。装入终止操作不重建索引，因此会忽略 USE <tablespace-name> 选项。

在装入操作的构建阶段，将在系统临时表空间中构建索引。在索引复制阶段，将索引从系统临时表空间复制至原始表空间。为确保原始索引表空间中有足够的空间来容纳

新的索引，在构建阶段应在原始表空间中分配空间。因此，如果装入操作用完索引空间，在构建阶段它就应该在原始表空间中分配空间。如果执行了以上操作，那么原始索引就不会丢失。

索引复制阶段在构建和删除阶段之后进行。在索引复制阶段开始之前，表以独占方式锁定。即，在整个索引复制阶段它不能用于读访问。因为索引复制阶段是物理复制，所以该表可能有很长一段时间不可用。

注: 如果系统临时表空间或索引表空间的其中任一表空间是 DMS 表空间，那么从系统临时表空间读取可能导致对系统临时表空间进行随机 I/O，并且可能导致延迟。写入索引表空间时仍然使用 DISK_PARALLELISM 值进行优化。

大型索引的注意事项

要提高在装入期间构建大型索引时的性能，调整 *sortheap* 数据库配置参数很有用。*sortheap* 分配专门用于在装入操作期间对索引键进行排序的内存量。例如，要引导 load 实用程序对每个索引使用 4000 页主存储器来进行键排序，将 *sortheap* 设置为 4000 页、断开所有应用程序与数据库的连接，然后发出 LOAD 命令。

如果索引很大以致不能在内存中排序，那么会出现*排序溢出*。即，数据将分割为若干次“排序运行”并且存储在日后合并的临时表空间中。使用 sort_overflows 监视元素来确定是否出现了排序溢出。如果无法通过增大 *sortheap* 参数的大小来避免排序溢出，那么应确保临时表空间的缓冲池足够大，以便将溢出导致的磁盘 I/O 量降至最低。而且，为了在排序运行合并期间获得 I/O 并行性，建议使用多个容器（每个容器都在不同的磁盘设备上）来声明临时表空间。如果对一个表定义了多个索引，那么因为装入操作会将所有键保存在内存中，所以内存消耗将按比例地增加。

延迟创建索引

通常，通过指定 REBUILD 或 INCREMENTAL 方式允许在装入操作期间创建索引比延迟创建索引的效率要高。如图 11 所示，通常用三个步骤来构建表: 装入数据、构建索引和收集统计信息。这会导致装入操作、索引创建（每个表可能有多个索引）和统计信息收集（导致对表数据和所有索引的 I/O）期间发生大量数据 I/O。如果让 load 实用程序通过传递一次数据完成所有任务，那么处理速度会快得多。然而，您应该注意，如果遇到重复项，那么唯一索引会降低装入性能。

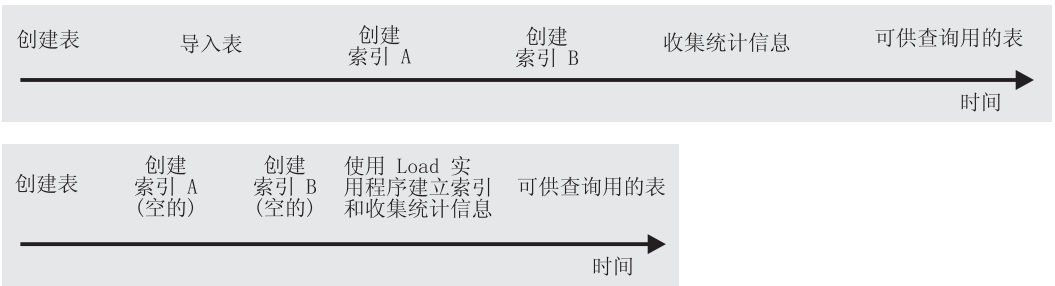


图 11. 通过建立并行索引和统计信息收集来提高装入性能. 通常用三个步骤来建立表: 数据装入、索引构建和统计信息收集。这会导致装入操作、索引创建（每个表可能有多个索引）和统计信息收集（导致对表数据和所有索引的 I/O）期间发生大量数据 I/O。如果让 load 实用程序通过传递一次数据完成所有任务，那么处理速度会快得多。

在某些情况下，延迟创建索引并调用 CREATE INDEX 语句可以提高性能。在索引重建期间进行排序将使用多达 *sortheap* 页。如果需要更多空间，那么会使用 TEMP 缓冲池并（最终）溢出至磁盘。如果装入溢出，并因此而导致性能下降，那么可能最好是将

LOAD 与 INDEXING MODE DEFERRED 配合运行，然后再重新创建索引。CREATE INDEX 一次将创建一个索引，在多次扫描表来搜索键的同时还降低了内存使用量。

使用 CREATE INDEX 语句而不同时使用装入操作来构建索引的另一个优点是，如果打开了 INTRA PARALLEL，那么 CREATE INDEX 语句可以使用多个进程或线程来对键进行排序。实际构建索引时并不是并行执行的。

在装入操作期间创建压缩字典

符合某个条件的装入插入操作和装入替换操作会触发自动字典创建（ADC）。处理足够多数据后，如果对启用了 COMPRESS 属性的表执行装入操作并且压缩字典不存在，那么会进行 ADC。

数据行压缩使用基于静态字典的压缩算法来压缩数据。但是，表中必须先存在字典才能进行压缩。在装入操作期间，缺省行为（由 KEEPDICTIONARY 选项指示）是遵循现有字典，或者如果字典不存在，那么在扫描了某个阈值的数据（大约 1MB）后生成字典。假设目标表中的数据表示将存储在该表中的数据的类型，load 实用程序使用这些数据来构建字典。如果目标表中预先存在的数据不够，在 load 实用程序采样了足够多的输入数据后，它将使用这些输入数据和预先存在的数据来构建字典。

在范围分区表上进行 ADC 时，每个分区都被视为一个单独的表。将没有任何跨分区字典。已经包含字典的分区上不会进行 ADC，并且为每个分区生成的字典仅基于该分区中预先存在的数据（在需要时还基于已装入的数据）。

在 KEEPDICTIONARY 行为后面会隐式执行任何 INSERT 方式的装入操作。但是，对于 load REPLACE 操作，您具有其他选项：RESETDICTIONARY 选项。

使用 KEEPDICTIONARY 选项的 load REPLACE

只要目标表启用了 COMPRESS 属性，使用 KEEPDICTIONARY 选项的 load REPLACE 就会保留现有字典并使用它来压缩已装入的数据。如果字典不存在，那么 load 实用程序将为启用了 COMPRESS 属性的表生成新字典（只要装入到表中的数据超过 1 MB 阈值）。由于要替换目标表中的数据，所以 load 实用程序仅使用输入数据来构建字典。在创建字典后，它将插入到表中并且装入操作继续。

使用 RESETDICTIONARY 选项的 load REPLACE

装入到打开了 COMPRESS 属性的表中时，使用 RESETDICTIONARY 选项有两种主要的含义。首先，在 load REPLACE 完成后，只要目标表中存在任意数量的数据，就会进行 ADC。也就是说，新的压缩字典可以基于单个数据行。另一种含义是如果满足下列任何情况，那么将删除现有字典而不是替换它（目标表不再具有压缩字典）：

- 对关闭了 COMPRESS 属性的表执行该操作
- 未装入任何内容（零行），在这种情况下 ADM5591W 将打印到通知日志

注：如果在执行使用 RESETDICTIONARY 选项的 load REPLACE 后发出 load TERMINATE 操作，那么将会删除现有压缩字典，而不是将其替换掉。

性能影响

由于下列原因，ADC 影响装入操作的性能：

- 表数据的初始扫描
  对于装入插入操作，在构建压缩字典之前将扫描所有预先存在的表数据，而不仅仅扫描 ADC 的最小阈值。因此，随着表大小的增大，进行这种扫描所用的时间也增加。
- 构建压缩字典所需的其他处理
  实际用于构建字典的时间很少。此外，缺省情况下，在构建字典后，ADC 就关闭了。

用于提高装入性能的选项

可以使用各种命令参数来优化装入性能。还有许多对装入来说唯一的文件类型修饰符，在某些情况下，这些修饰符可以极大地提高该实用程序的性能。

命令参数

如果用户未指定 DISK_PARALLELISM、CPU_PARALLELISM 和 DATA BUFFER 参数的值，那么 load 实用程序将尝试通过确定这些参数的最优值来获得最好的性能。根据实用程序堆大小及可用空间来进行优化。在尝试调整这些参数以满足特殊需要之前，考虑使用自主 DISK_PARALLELISM 和 CPU_PARALLELISM 设置。

以下是有关 load 实用程序提供的各种选项所带来的性能影响的信息:

ALLOW READ ACCESS

此选项允许您在进行装入操作时查询表。只能查看装入操作之前表中已存在的数据。如果还指定了 INDEXING MODE INCREMENTAL 选项，并且装入操作失败，那么后续装入终止操作可能必须校正索引中的不一致。这将需要涉及大量 I/O 的索引扫描。如果还对装入终止操作指定了 ALLOW READ ACCESS 选项，那么会将缓冲池用于 I/O。

COPY YES 或 COPY NO

使用此参数来指定在装入操作期间是否创建输入数据的副本。仅当启用了正向恢复时，COPY YES 才适用，并且由于装入操作期间会复制所有装入数据，所以使用此参数会降低装入性能。I/O 活动增加可能会导致 I/O 绑定系统上的装入时间增加。如果指定多个设备或不同磁盘上的多个目录，那么可能会因为此操作而使性能受到影响。仅当启用了正向恢复时，COPY NO 才适用，并且它不会影响装入性能。但是，所有与已装入的表相关的表空间将处于“备份暂挂”状态，并且必须先备份这些表空间才能访问该表。

CPU_PARALLELISM

借助此参数来使用分区内并行性（如果机器具有此功能）从而大幅改进装入性能。该参数指定 load 实用程序用于分析、转换、格式化数据记录的进程或线程的数目。允许的最大数目是 30。如果内存不足以支持指定值，那么实用程序将调整该值。如果未指定此参数，那么 load 实用程序将根据系统上的 CPU 数目选择缺省值。

只要满足下列条件，无论此参数的值如何，都将保留源数据中的记录顺序（请参阅第 147 页的图 12）:

- 未指定 `anyorder` 文件类型修饰符
- 未指定 PARTITIONING_DBPARTNUMS 选项（并且将多个分区用于分区）

如果表包括 LOB 或 LONG VARCHAR 数据，那么 CPU_PARALLELISM 将设置为 1。在这种情况下不支持并行性。

尽管此参数并未限制为只能供对称多处理器（SMP）硬件使用，但在非 SMP 环境中使用它在性能方面也没什么太大益处。

图 12. 在装入操作期间使用分区内并行性时，将会保留源数据中的记录顺序。

DATA BUFFER

DATA BUFFER 参数指定分配给 load 实用程序以用作缓冲区的内存总量（以 4KB 为单位）。建议此缓冲区在大小上等于若干*扩展数据块*。数据缓冲区将从实用程序堆分配。根据系统上可用的存储量，应考虑分配更多内存以供 DB2 实用程序使用。可相应修改数据库配置参数 *util_heap_sz*（实用程序堆大小）。*util_heap_sz* 的缺省值为 5 000 4 KB 的页。因为 load 实用程序只是使用实用程序堆内存的若干实用程序的其中一个，所以建议将此参数定义的不超过 50% 的页供 load 实用程序使用，并且将实用程序堆定义得足够大。

DISK_PARALLELISM

DISK_PARALLELISM 参数指定 load 实用程序用来将数据记录写至磁盘的进程或线程数。借助此参数在装入数据时使用可用容器，从而大幅改进装入性能。允许的最大数目是 CPU_PARALLELISM 值（load 实用程序使用的实际量）的四倍或 50 中较大的数字。缺省情况下，DISK_PARALLELISM 等于包含对其装入表的对象的所有表空间中的表空间容器的总和，但此值超过允许的最大值时除外。

NONRECOVERABLE

如果启用了正向恢复，那么在前滚后不需要对表恢复装入事务的情况下使用此参数。NONRECOVERABLE 装入和 COPY NO 装入具有完全相同的性能。但是，在潜在数据丢失方面却有重大差别。NONRECOVERABLE 装入将表标记为不可前滚恢复，并同时使得能够完全访问表。这可能会产生一个问题，在需要前滚装入操作的情况下，已装入的数据以及所有对表的后续更新都会丢失。COPY NO 装入使所有从属表空间处于“备份暂挂”状态，这将导致在执行备份之前，表不可访问。因为在该类型的装入后会强制您执行备份，所以您不存在丢失已装入的数据或对表的后续更新的风险。也就是说，COPY NO 装入完全可恢复。

注: 如果在后续复原和前滚恢复操作期间遇到这些装入事务，那么该表将不会更新，并且被标记为 `invalid`。将忽略对此表的进一步操作。在前滚操作完成后，只能废弃该表。

SAVECOUNT

使用此参数来设置在装入操作的**装入阶段期间**建立一致点的时间间隔。为建立一致点而执行活动同步需要花一些时间。如果进行得太频繁，装入性能会大幅下降。如果要装入大量行，那么建议您指定较大的 SAVECOUNT 值（例如，在涉及 1 亿条记录的装入操作中指定值 10000000）。

只要装入重新启动操作从装入阶段恢复，该操作就会从上一个一致点自动继续。

STATISTICS USE PROFILE

收集表统计信息概要文件中指定的统计信息。即使装入操作本身的性能下降

（特别是在指定 DETAILED INDEXES ALL 时），与在完成装入操作后调用 RUNSTATS 实用程序相比，使用此参数来收集数据分发和索引统计信息更有效。

为优化性能，应用程序需要尽可能最佳的数据分发和索引统计信息。一旦更新统计信息，应用程序就可以根据最新的统计信息使用新的表数据访问路径。可通过使用 BIND 命令重新绑定应用程序包来创建新的表访问路径。通过运行带有 SET PROFILE 选项的 RUNSTATS 命令来创建表统计信息概要文件。

将数据装入到大表中时，建议对 *stat_heap_sz*（统计信息堆大小）数据库配置参数指定较大的值。

USE <tablespace-name>

如果正在执行 ALLOW READ ACCESS 装入并且建立索引方式为 REBUILD，那么此参数允许在系统临时表空间中重建索引，并在装入操作的索引复制阶段将其复制回索引表空间。

缺省情况下，将在原始索引所在的表空间中构建完全重建的索引（也称为*影子索引*）。因为原始索引和影子索引同时位于同一表空间中，所以这可能会导致资源问题。如果影子索引与原始索引是在同一个表空间中构建的，那么影子索引将瞬时替换原始索引。但是，如果影子索引是在系统临时表空间中构建的，那么装入操作需要索引复制阶段，该阶段会将索引从系统临时表空间复制至索引表空间。复制阶段将涉及相当多的 I/O。如果其中任一表空间是 DMS 表空间，那么系统临时表空间的 I/O 可能不是顺序进行的。在索引复制阶段将使用 DISK_PARALLELISM 选项指定的值。

WARNINGCOUNT

使用此参数来指定强制装入操作终止之前该实用程序可返回的警告数目。如果您只需要很少警告或不需要警告，那么将 WARNINGCOUNT 参数设置为相对较小的数字。装入操作将在达到 WARNINGCOUNT 数目时停止。这允许您在尝试完成装入操作之前解决问题。

文件类型修饰符

ANYORDER

缺省情况下，load 实用程序将保留源数据的记录顺序。在 SMP 环境中进行装入时，要求并行处理之间保持同步以确保保留该顺序。

在 SMP 环境中，指定 `anyorder` 文件类型修饰符将指示 load 实用程序不保留顺序，由于这样做不必执行保留该顺序所需的同步，所以将会提高性能。但是，如果要装入的数据进行了预先排序，那么 `anyorder` 可能会破坏预先排好的顺序，使得后续查询也无法受益于预先排序。

注: 如果 CPU_PARALLELISM 为 1，那么 `anyorder` 文件类型修饰符不起作用，并且它与 SAVECOUNT 选项不兼容。

BINARYNUMERICS、ZONEDDECIMAL 和 PACKEDDECIMAL

对于固定长度的非定界 ASCII（ASC）源数据，用二进制表示数字数据可能会提高装入时的性能。如果指定了 `packeddecimal` 文件类型修饰符，那么 load 实用程序会使用压缩十进制格式（每个字节占两位）表示十进制数据。如果指定了 `zoneddecimal` 文件类型修饰符，那么 load 实用程序会使用分区十进制格式（每个字节占一位）表示十进制数据。对于所有其他数字类型，如果指定了 `binarynumerics` 文件类型修饰符，那么 load 实用程序会使用二进制格式表示数据。

注:

- 在指定了 binarynumerics、packeddecimal 或 zoneddecimal 文件类型修饰符时，无论使用什么平台，都使用大尾数法（高字节在前面）格式表示数字数据。
- packeddecimal 和 zoneddecimal 文件类型修饰符互斥。
- packeddecimal 和 zoneddecimal 文件类型修饰符仅适用于十进制目标列，并且二进制数据必须与目标列定义匹配。
- 在指定了 binarynumerics、packeddecimal 或 zoneddecimal 文件类型修饰符时，必须指定 reclen 文件类型修饰符。

FASTPARSE

使用时务必小心谨慎。如果知道要装入的数据有效，那么没必要让装入像对较可疑的数据那样执行那么多的语法检查。事实上，缩小语法检查的范围可以将装入性能提高大约 10% 或 20%。这可以通过使用 fastparse 文件类型修饰符来实现，该修饰符可以减少对 ASC 和 DEL 文件中用户提供的列值执行的数据检查。

NOROWWARNINGS

在装入操作期间，关于已拒绝的行的警告消息将写入指定的文件中。但是，如果 load 实用程序必须处理大量已拒绝的、无效或已截断的记录，那么可能会对装入性能产生负面影响。如果预计到会产生许多警告，那么使用 norowwarnings 文件类型修饰符来抑制记录这些警告很有用。

PAGEFREESPACE、INDEXFREESPACE 和 TOTALFREESPACE

随着时间的推移，表中插入和更新的数据不断增加，重组表和索引的需求也就更迫切。一种解决方案是使用 pagefreespace、indexfreespace 和 totalfreespace 来增大用于表和索引的可用空间量。前两个修饰符优先于 PCTFREE 值，它们指定要作为可用空间保留的数据和索引页数的百分比，而 totalfreespace 指定要作为可用空间追加至表的总页数的百分比。

用于维护引用完整性的装入功能

虽然 load 实用程序通常比 Import 实用程序的效率更高，但它需要许多功能来确保要装入的信息的引用完整性:

- **表锁定**，它在装入操作期间提供并行控制并防止进行不受控数据访问
- **表状态**和**表空间状态**，它可以控制对数据的访问或引发特定用户操作
- **装入异常表**，它确保不会在您不知道的情况下简单地删除无效数据行

在执行装入操作后检查完整性违例

装入操作完成后，如果下列任何一个条件成立，装入的表就会处于 READ 或 NO ACCESS 方式下的设置完整性暂挂状态:

- 对该表定义了表检查约束或引用完整性约束。
- 该表包含生成列，启动装入操作时使用的是版本 7 或更旧版本的客户机。
- 该表有引用该表的派生立即具体化查询表或派生立即登台表。
- 该表是登台表或具体化查询表。

与装入的表对应的 SYSCAT.TABLES 条目的 STATUS 标志指示了该表的设置完整性暂挂状态。要让装入的表完全可用，STATUS 值必须为 N，ACCESS MODE 值必须为 F，这表示该表完全可访问并处于正常状态。

如果装入的表带有派生表，那么可以指定 SET INTEGRITY PENDING CASCADE 参数以指示是否应该立即将装入的表的设置完整性暂挂状态级联到派生表。

如果装入的表带有约束以及派生外键表、从属具体化查询表和从属登台表，并且所有这些表在装入操作前都处于正常状态，那么根据指定的装入参数，将产生以下结果:

INSERT、ALLOW READ ACCESS 和 SET INTEGRITY PENDING CASCADE IMMEDIATE

装入的表、其从属具体化查询表以及从属登台表将处于设置完整性暂挂状态，并允许进行读访问。

INSERT、ALLOW READ ACCESS 和 SET INTEGRITY PENDING CASCADE DEFERRED

只有装入的表才会处于设置完整性暂挂状态，并且允许进行读访问。派生外键表、派生具体化查询表和派生登台表将保持它们的原始状态。

INSERT、ALLOW NO ACCESS 和 SET INTEGRITY PENDING CASCADE IMMEDIATE 装入的表、其从属具体化查询表以及从属登台表将处于设置完整性暂挂状态，并且不允许进行访问。

INSERT 或 REPLACE、ALLOW NO ACCESS 和 SET INTEGRITY PENDING CASCADE DEFERRED

只有装入的表才会处于设置完整性暂挂状态，并且不允许进行访问。派生外键表、派生立即具体化查询表和派生立即登台表将保持它们的原始状态。

REPLACE、ALLOW NO ACCESS 和 SET INTEGRITY PENDING CASCADE IMMEDIATE

该表以及其所有派生外键表、派生立即具体化查询表和派生立即登台表都将处于设置完整性暂挂状态，并且不允许进行访问。

注: 在装入替换操作中指定 ALLOW READ ACCESS 选项将导致错误。

要脱离设置完整性暂挂状态，请使用 SET INTEGRITY 语句。SET INTEGRITY 语句检查表的约束违例情况并使该表脱离设置完整性暂挂状态。如果所有装入操作都是以 INSERT 方式执行的，那么可使用 SET INTEGRITY 语句来以递增方式处理约束（即，它将只检查表的追加部分是否存在约束违例情况）。例如:

```
db2 load from infile1.ixf of ixf insert into table1
db2 set integrity for table1 immediate checked
```

只检查 TABLE1 的追加部分是否存在约束违例情况。与检查整个表相比，只检查追加部分是否存在约束违例情况速度更快，对于包含少量追加数据的大型表来说尤其如此。

如果装入表时指定了 SET INTEGRITY PENDING CASCADE DEFERRED 选项，并且使用了 SET INTEGRITY 语句来检查完整性违例，那么派生表将处于设置完整性暂挂状态，并且不允许进行访问。要使那些表脱离此状态，必须发出显式的请求。

如果使用 INSERT 选项来装入带有从属具体化查询表或从属登台表的表，并且使用了 SET INTEGRITY 语句来检查完整性违例，那么该表将脱离设置完整性暂挂状态并处于“无数据移动”状态。这样做是为了便于以后对从属具体化查询表进行递增刷新以及对从属登台表进行递增传播。在“无数据移动”状态下，不允许执行可能会在表中移动行的操作。

可以通过在发出 SET INTEGRITY 语句时指定 FULL ACCESS 选项来覆盖“无数据移动”状态。该表将完全可访问，但以后执行 REFRESH TABLE 语句时将完全重新计算从属具体化查询表，并且将强制使从属登台表处于不完整状态。

如果执行装入操作时指定了 ALLOW READ ACCESS 选项，那么在使用 SET INTEGRITY 语句检查约束违例之前，该表将一直处于读访问状态。一旦落实装入操作，应用程序就能够查询在执行该装入操作前存在的表数据，但在发出 SET INTEGRITY 语句前，应用程序无法查看新装入的数据。

在检查约束违例前，可以对表执行多次装入操作。如果所有装入操作都以 ALLOW READ ACCESS 方式完成，那么只有执行第一次装入操作前在表中存在的数据可供查询。

调用一次此语句可以检查一个或多个表。如果要单独地检查从属表，那么父表不能处于设置完整性暂挂状态。否则，必须同时检查父表和从属表。对于一个引用完整性周期来说，必须在 SET INTEGRITY 语句的单一调用中包括该周期内涉及到的所有表。这样就可以在装入从属表时方便地检查父表的约束违例情况。仅当两个表不在同一表空间中时，才会发生这种情况。

发出 SET INTEGRITY 语句时，可以指定 INCREMENTAL 选项以显式请求递增式处理。在大多数情况下，由于 DB2 数据库将选择递增式处理，所以不需要指定此选项。如果不可能进行递增式处理，就会自动采用完全处理方法。当指定了 INCREMENTAL 选项，但不可能进行递增式处理时，在下列情况下，将返回错误:

- 当该表处于设置完整性暂挂状态时，已对该表添加新约束。
- 在上次对该表执行完整性检查之后，已执行装入替换操作，或者已激活 NOT LOGGED INITIALLY WITH EMPTY TABLE 选项。
- 已经以非递增方式对父表执行了装入替换操作或完整性检查操作。
- 该表在迁移前处于设置完整性暂挂状态。在迁移后第一次检查表的完整性时，需要进行完全处理。
- 包含该表或其父表的表空间已前滚到某个时间点，并且该表及其父表在不同的表空间中。

如果一个表在 SYSCAT.TABLES 目录的 CONST_CHECKED 列中有一个或多个 W 值，并且在 SET INTEGRITY 语句中未指定 NOT INCREMENTAL 选项，就会以递增方式处理该表，并且将把 SYSCAT.TABLES 的 CONST_CHECKED 列标记为 U 以指示系统并未验证所有数据。

SET INTEGRITY 语句不会由于删除违反约束的行而激活任何 DELETE 触发器，但一旦使该表脱离设置完整性暂挂状态，触发器就会处于活动状态。因此，如果更正数据并将异常表中的行插入到装入的表中，就会激活对该表定义的任何 INSERT 触发器。您应该考虑这种情况的影响。可以选择删除 INSERT 触发器，从异常表插入行，然后重新创建 INSERT 触发器。

使用 SET INTEGRITY 检查约束违例

通常，在下列三种情况下需要对表手动执行完整性处理：在将数据装入表以后；当通过对表添加约束改变表时；以及当改变表以添加生成的列时。

- 要打开对表的约束检查并执行对该表的完整性处理，您需要具有下列其中一项权限或特权:

- SYSADM 或 DBADM 权限
- 对检查的表的 CONTROL 特权，及对异常表的 INSERT 特权（如果异常被记录到一个或多个表中的话）
- 对那些通过语句隐式设置为“设置完整性暂挂”状态的所有派生外键表、派生立即具体化查询表和派生立即登台表的 CONTROL 特权。
- LOAD 权限，并且如果异常被记录到一个或多个表中的话:
  - 对所检查的每个表的 SELECT 和 DELETE 特权
  - 对异常表的 INSERT 特权

• 要打开对表的约束检查而不执行对该表的完整性处理，您需要下列其中一项权限或特权:
- SYSADM 或 DBADM 权限
- 对所检查的表的 CONTROL 特权
- 对那些通过语句隐式设置为“设置完整性暂挂”状态的所有派生外键表、派生立即具体化查询表和派生立即登台表的 CONTROL 特权

• 要关闭表的约束检查、立即刷新或立即传播，您需要具有下列其中一项权限或特权:
- SYSADM 或 DBADM 权限
- 对通过语句关闭其完整性检查的表、所有派生外键表、派生立即具体化查询表和派生立即登台表的 CONTROL 特权
- LOAD 权限

如果一个表定义了约束或者具有从属外键表、从属具体化查询表或从属登台表，那么装入操作将导致该表自动进入“设置完整性暂挂”状态。当装入操作完成时，可以验证装入的数据的完整性，并且可以对表打开约束检查。如果一个表具有从属外键表、从属具体化查询表或从属登台表，那么它们将自动进入“设置完整性暂挂”状态。将需要使用“设置完整性”窗口来对这些表中的每个表单独执行完整性处理。

如果正在通过添加外键、检查约束或生成列来改变一个表，那么需要在改变该表前关闭约束检查。在添加约束后，需要检查现有数据是否违反了新添加的约束，并且需要重新打开约束检查。另外，如果正在将数据装入表中，那么要等到完成装入后才能对该表激活约束检查。如果正在将数据导入表中，那么应在导入前对表激活约束检查。

约束检查指的是检查约束违例、外键违例和生成列违例。完整性处理指的是除了执行约束检查外，还填充标识列和生成列、刷新具体化查询表以及传播登台表。

通常，自动强制执行对表的引用完整性和检查约束，立即自动刷新具体化查询表，并且自动传播登台表。在某些情况下，可能需要手动更改此行为。

要使用控制中心来检查约束违例:

1. 打开“设置完整性”窗口：从“控制中心”中展开对象树，直到找到**表**文件夹为止。单击**表**文件夹。所有现有表都会显示在窗口右边的窗格中。右键单击想要的表并从弹出菜单中选择**设置完整性**。“设置完整性”窗口将打开。
2. 检查您正在处理的表的“当前完整性状态”。
3. 要打开对表的约束检查而不检查表数据:
   a. 选择**立即并且未检查**单选按钮。
   b. 指定要打开的完整性处理的类型。

c. 选择**完全访问**单选按钮以立即对表执行数据移动操作（例如，重组或重新分发）。但是应注意，后续刷新从属具体化查询表将花更长时间。如果表具有相关联的具体化查询表，那么建议您不要选择此单选按钮，以减少刷新具体化查询表所需的时间。

4. 要打开对表的约束检查并检查现有表数据:
   a. 选择**立即并且已检查**单选按钮。
   b. 选择要执行的完整性处理的类型。如果**当前完整性状态**显示具体化查询表的约束检查值不完整，那么不能以增量方式刷新具体化查询表。
   c. 可选: 如果想要在完整性处理期间填充标识列或生成列，那么请选择**强制生成**复选框。
   d. 如果表不是登台表，那么请确保未选择**修剪**复选框。
   e. 选择**完全访问**单选按钮以立即对表执行数据移动操作。
   f. 可选: 指定异常表。违反引用或检查约束的任何行都将从您的表删除并复制至异常表中。如果未指定异常表，那么当违反了约束时，仅将检测到的第一个违例返回给您且表将处于“设置完整性暂挂”状态。
5. 要关闭对表的约束检查、立即刷新或立即传播:
   a. 选择**关闭**单选按钮。表将处于“设置完整性暂挂”状态。
   b. 使用**级联**选项来指定是想要立即级联还是延迟级联。如果正在进行立即级联，那么使用**具体化查询表**、**外键表**和**登台表**复选框来指示想要级联的表。

      注: 如果关闭对父表的约束检查，并指定想要将更改级联至外表键，那么还将关闭它的所有派生外键表的外键约束。如果关闭对基础表的约束检查，并指定想要将检查暂挂状态级联至具体化查询表，那么还将关闭它的所有从属具体化查询表的立即刷新属性。如果关闭对基础表的约束检查，并指定想要将“设置完整性暂挂”状态级联至登台表，那么还将关闭它的所有从属登台表的立即传播属性。

要使用命令行来检查约束违例，可使用 SET INTEGRITY 语句。

故障诊断提示

症状 当尝试打开对表的约束检查、立即刷新或立即传播时，接收到下列错误消息:

DB2 消息

当父表或基础表 TABLE2 处于“设置完整性暂挂”状态或将使用 SET INTEGRITY 语句使它处于“设置完整性暂挂”状态时，不能使用 SET INTEGRITY 语句检查从属表 TABLE1。

其中，TABLE1 是您正在尝试对其打开约束检查、立即刷新或立即传播的表，并且它从属于 TABLE2。

可能的原因

对于其父表或基础表处于“设置完整性暂挂”状态的表，不能打开约束检查、立即刷新或立即传播。

操作 通过对该表打开约束检查使其父表或基础表脱离“设置完整性暂挂”状态。从 DB2 消息中标识为父表或基础表的表开始。如果该表从属于另一表，必须以自顶向下分析法从位于相关链顶端的表打开约束检查。

注意: 如果选择的表与一个或多个表具有循环引用约束关系，那么不能使用“设置完整性”窗口来打开约束检查。在这种情况下，必须使用“命令编辑器”来发出 SQL SET INTEGRITY 命令。

在装入操作期间进行表锁定

在大多数情况下，load 实用程序使用表级别锁定功能来限制对表的访问。锁定级别取决于装入操作所处的阶段以及是否已将装入操作指定为允许读访问。

ALLOW NO ACCESS 方式的装入操作在装入期间对表使用超级互斥锁定（Z 锁定）。

在 ALLOW READ ACCESS 方式的装入操作开始之前，load 实用程序将等待所有在该装入操作前开始的应用程序释放对目标表的锁定。在装入操作开始时，load 实用程序对表获取更新锁定（U 锁定）。它在落实数据前将一直挂起这个锁定。当 load 实用程序对表获取 U 锁定时，它将等待所有在装入操作开始前对该表挂起了锁定的应用程序释放那些锁定，即使它们使用的是兼容锁定亦如此。这是通过临时地将 U 锁定升级为 Z 锁定实现的，只要对目标表请求的锁定与装入操作的 U 锁定兼容，这种锁定就不会与新请求的表锁定发生冲突。在落实数据时，load 实用程序将把该锁定升级为 Z 锁定。因此，在落实时会出现一定程度的延迟，这是由于 load 实用程序等待使用冲突锁定的应用程序完成而导致的。

注: 在进行装入前，装入操作在等待应用程序释放对表的锁定时可能会超时。但是，装入操作在等待获取落实数据所需的 Z 锁定时不会超时。

使用冲突锁定的应用程序

使用 LOAD 命令的 LOCK WITH FORCE 选项来强制应用程序释放对目标表的冲突锁定，以使装入操作能够继续执行。应先强制使挂起了下列锁定的应用程序释放锁定，然后 ALLOW READ ACCESS 方式的装入操作才能继续执行。

- 与表更新锁定有冲突的表锁定（例如，导入应用程序或插入应用程序挂起的表锁定）。
- 在装入操作的落实阶段存在的所有表锁定。

load 实用程序不会强制使对系统目录表挂起冲突锁定的应用程序释放锁定。如果 load 实用程序强制某个应用程序脱离系统，那么该应用程序就会丢失其数据库连接并返回错误（SQL1224N）。

对可恢复数据库执行的装入操作指定 COPY NO 选项时，将以共享方式锁定目标表空间中的所有对象，然后使该表空间处于“备份暂挂”状态。无论使用哪种访问方式，都会发生这种情况。如果指定了 LOCK WITH FORCE 选项，并且有应用程序对表空间中的对象挂起的锁定与共享锁定发生冲突，那么会强制所有那些应用程序释放锁定。

读访问装入操作

load 实用程序提供了两个选项来控制其他应用程序对正在装入的表的访问程度。ALLOW NO ACCESS 选项以独占方式锁定表，在装入该表时不允许对表数据进行访问。

ALLOW NO ACCESS 选项是缺省行为。ALLOW READ ACCESS 选项不允许其他应用程序对该表进行任何写访问，但允许对预先存在的数据进行读访问。本节描述 ALLOW READ ACCESS 选项。

在装入操作执行过程中，在装入操作启动前存在的表数据和索引数据对查询可视。请考虑以下示例:

1. 创建包含一个整数列的表:

```
create table ED (ed int)
```

2. 装入三行:

```
load from File1 of del insert into ED
...
读取的行数                      = 3
跳过的行数          = 0
已装入的行数                      = 3
拒绝的行数          = 0
删除的行数          = 0
已落实的行数                      = 3
```

3. 查询该表:

```
select * from ED

ED
-----------
                    1
                    2
                    3

  3 record(s) selected.
```

4. 在指定了 ALLOW READ ACCESS 选项的情况下执行装入操作并装入另外两行数据:

```
load from File2 of del insert into ED allow read access
```

5. 同时，使用另一个连接，在装入操作执行过程中查询该表:

```
select * from ED

ED
-----------
                    1
                    2
                    3

  3 record(s) selected.
```

6. 等待装入操作完成，然后查询该表:

```
select * from ED

ED
-----------
                    1
                    2
                    3
                    4
                    5

  5 record(s) selected.
```

由于 ALLOW READ ACCESS 选项允许用户在任何时间（甚至在装入操作执行时或者在装入操作失败后）访问表数据，所以，装入大量数据时，此选项非常有用。在 ALLOW READ ACCESS 方式下，装入操作的行为独立于应用程序的隔离级别。即，具有任何隔离级别的阅读器始终能够读取预先存在的数据，但它们在装入操作完成前无法读取新装入的数据。

在装入操作的整个执行过程中，除了操作开始和操作结束时以外，都允许进行读访问。

首先，在设置阶段接近结束时，装入操作将获取特殊的 Z 锁定并占用一小段时间。如果某个应用程序在装入操作请求这个特殊的 Z 锁定前对该表挂起了不兼容的锁定，装入操作在发生超时和失败之前将等待一段有限的时间以允许这个不兼容的锁定被释放。等待时间长度由 *locktimeout* 数据库配置参数确定。如果指定了 LOCK WITH FORCE 选项，装入操作就会强制其他应用程序释放锁定以避免超时。装入操作获取特殊的 Z 锁定、落实设置阶段、释放该锁定并进入装入阶段。在 ALLOW READ ACCESS 方式的装入操作启动后，任何对该表请求读访问锁定的应用程序都将获得该锁定，而不会与这个特殊的 Z 锁定发生冲突。尝试读取目标表中现有数据的新应用程序也能够成功地完成操作。

其次，在装入操作结束时，load 实用程序在落实数据之前对该表获取互斥锁定（Z 锁定）。load 实用程序将等待所有对该表挂起了锁定的应用程序释放那些锁定。这会导致落实数据前发生延迟。LOCK WITH FORCE 选项用来强制有冲突的应用程序释放锁定，这样装入操作就能够继续执行而不必等待。通常，ALLOW READ ACCESS 方式的装入操作获取互斥锁定并占用一小段时间；但是，如果指定了 USE <tablespace-name> 选项，就会在整个索引复制阶段占用互斥锁定。

对在多个数据库分区上定义的表运行 load 实用程序时，装入进程技术模型将在每个单独的数据库分区上执行，这意味着独立于其他数据库分区获取并释放了锁定。因此，如果同时执行查询或其他操作并争用相同锁定，那么就可能会出现死锁。例如，假定操作 A 在数据库分区 0 上被授予表锁定，并且装入操作在数据库分区 1 上被授予表锁定。由于操作 A 正在等待被授予数据库分区 1 上的表锁定，而装入操作正在等待数据库分区 0 上的表锁定，所以可能出现死锁。在这种情况下，死锁检测器将任意回滚其中一个操作。

注:

1. 如果装入操作中断或失败，它将保持发出装入操作时指定的访问级别。也就是说，如果 ALLOW NO ACCESS 方式的装入操作失败，那么在发出装入终止或装入重新启动操作之前将无法访问表数据。如果 ALLOW READ ACCESS 方式的装入操作异常终止，那么仍然能够对预先存在的表数据进行读访问。
2. 如果对中断或失败的装入操作指定了 ALLOW READ ACCESS 选项，那么也可以对装入重新启动或装入终止操作指定此选项。但是，如果中断或失败的装入操作指定了 ALLOW NO ACCESS 选项，那么不能对装入重新启动或装入终止操作指定 ALLOW READ ACCESS 选项。

在下列情况下，不支持 ALLOW READ ACCESS 选项:

- 指定了 REPLACE 选项。由于装入替换操作在装入新数据前将截断现有表数据，因此，在装入操作完成之前，没有预先存在的数据可供查询。
- 索引已被标记为无效并正在等待重建。在某些前滚情况下，可能会将索引标记为无效，也可以使用 db2dart 命令来达到此目的。
- 指定了 INDEXING MODE DEFERRED 选项。此方式将索引标记为需要重建。
- 正在重新启动或终止 ALLOW NO ACCESS 装入操作。在使表完全处于联机状态之前，不能对该表执行 ALLOW READ ACCESS 方式的装入操作。

- 对处于“设置完整性暂挂不访问”状态的表执行装入操作。对于在带有约束的表上执行的多个装入操作来说，情况亦如此。在发出 SET INTEGRITY 语句之前，不会使表处于联机状态。

通常，如果表数据处于脱机状态，在装入操作期间是无法进行读访问的，除非该表数据再次处于联机状态。

装入操作期间和之后的表空间状态

在装入操作期间，load 实用程序使用表空间状态来保持数据库一致性。这些状态通过控制对数据的访问或引发用户操作来起作用。

load 实用程序不会停顿（持久锁定）装入操作所使用的表空间，并且仅对指定了 COPY NO 参数的装入操作使用表空间状态。

可以使用 LIST TABLESPACES 命令来检查表空间状态。表空间可以同时处于多种状态。LIST TABLESPACES 命令返回的状态如下所示:

正常
“正常”状态是创建表空间后该表空间的初始状态，它指示当前没有（异常）状态影响表空间。

正在装入
“正在装入”状态指示正在表空间上进行装入。此状态不允许在装入操作期间备份从属表。“正在装入”表空间状态与“正在装入”表状态（所有装入操作中都使用此状态）不同，因为仅当对可恢复数据库指定了 **COPY NO** 参数时，load 实用程序才使表空间处于“正在装入”状态。表空间在装入操作持续期间将保持处于此状态。

备份暂挂
如果对可恢复数据库执行装入操作并且指定 **COPY NO** 参数，那么在第一次落实后表空间将处于“备份暂挂”表空间状态。不能更新处于“备份暂挂”状态的表空间。通过备份表空间即可使表空间脱离“备份暂挂”状态。由于装入操作开始时会更改表空间状态并且不能回滚，所以即使取消装入操作，表空间也保持处于“备份暂挂”状态。

复原暂挂
如果使用 **COPY NO** 选项成功执行了装入操作、复原数据库，然后前滚该操作，那么相关表空间将处于“复原暂挂”状态。要使表空间脱离“复原暂挂”状态，必须执行复原操作。

表空间状态的示例

如果将输入文件（staffdata.del）装入到 NEWSTAFF 表中，如下所示:

```
update db cfg for sample using logretain recovery;
backup db sample;
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff copy no;
connect reset;
```

并且打开另一个会话并发出下列命令:

```
connect to sample;
list tablespaces;
connect reset;
```

那么 USERSPACE1（样本数据库的缺省表空间）将处于“正在装入”状态，并且在第一次落实后，将处于“备份暂挂”状态。在装入操作完成后，LIST TABLESPACES 命令表明 USERSPACE1 现在处于“备份暂挂”状态:

```
  表空间标识                           = 2
名称                                 = USERSPACE1
类型                                 = 数据库管理的空间
内容                                 = 所有永久数据。大型表空间。
状态                                 = 0x0020
   详细说明:
备份暂挂
```

装入操作期间和之后的表状态

在装入操作期间，load 实用程序使用表状态来保持数据库一致性。这些状态通过控制对数据的访问或引发用户操作来起作用。

要确定表状态，发出 LOAD QUERY 命令，该命令还检查装入操作的状态。表可以同时处于多种状态。LOAD QUERY 命令返回的状态如下所示:

正常状态

“正常”状态是创建表后该表的初始状态，它指示当前没有（异常）状态影响表。

只读访问

如果指定了 ALLOW READ ACCESS 选项，那么表将处于“只读访问”状态。在调用 LOAD 命令前存在的表数据在装入操作运行期间可供只读访问。如果指定了 ALLOW READ ACCESS 选项并且装入操作失败，那么在装入操作前存在的表数据在故障发生后将继续可供只读访问。

正在装入

“正在装入”表状态指示正在表上进行装入。在装入操作成功完成后，load 实用程序将除去此瞬时状态。但是，如果装入操作失败或被中断，那么表状态将更改为“装入暂挂”。

正在重新分发

“正在重新分发”表状态指示正在表上进行重新分发。在 Redistribute 实用程序成功处理完表之后，该实用程序将除去此瞬时状态。但是，如果重新分发操作失败或被中断，那么表状态将更改为“重新分发暂挂”。

装入暂挂

“装入暂挂”表状态指示装入操作失败或被中断。可以执行下列其中一个步骤来除去“装入暂挂”状态:

- 找处故障原因。例如，如果 load 实用程序耗尽了磁盘空间，那么对表空间添加容器。然后，重新启动装入操作。
- 终止装入操作。
- 对装入操作失败时所处理的那个表运行 load REPLACE 操作。
- 使用最新的表空间或数据库备份，通过 RESTORE DATABASE 命令恢复所装入的表的表空间，然后执行进一步的恢复操作。

重新分发暂挂

“重新分发暂挂”表状态指示重新分发操作失败或被中断。可以执行 REDISTRIBUTE CONTINUE 或 REDISTRIBUTE ABORT 操作来除去“重新分发暂挂”状态。

不可重新启动装入

处于“不可重新启动装入”状态时，表已部分装入，并且不允许装入重新启动操作。在下面两种情况下，表会处于“不可重新启动装入”状态:

- 在未能成功地重新启动或终止的失败装入操作后，执行前滚操作
- 根据表处于“正在装入”或“装入暂挂”状态时创建的联机备份执行复原操作

表还将处于“装入暂挂”状态。要使表脱离“不可重新启动装入”状态，发出 LOAD TERMINATE 或 LOAD REPLACE 命令。

设置完整性暂挂

“设置完整性暂挂”状态指示已装入的表有未经验证的约束。当 load 实用程序开始对带有约束的表执行装入操作时，它就会使该表处于此状态。使用 SET INTEGRITY 语句以使该表脱离“设置完整性暂挂”状态。

1 类索引

“1 类索引”状态指示表当前使用已不推荐使用的 1 类索引。对那些索引使用 REORG 实用程序时，可以使用 CONVERT 选项将它们转换为 2 类索引。

不可用

通过不可恢复的装入操作执行前滚将使表处于“不可用”状态。处于此状态时，表不可用; 必须废弃该表或通过备份复原表。

处于多种状态的表的示例

如果将包含大量数据的输入文件（staffdata.del）装入到 NEWSTAFF 表中，如下所示:

```
connect to sample;
create table newstaff like staff;
load from staffdata.del of del insert into newstaff allow read access;
connect reset;
```

并且打开另一个会话并发出下列命令:

```
connect to sample;
load query table newstaff;
connect reset;
```

LOAD QUERY 命令将显示 NEWSTAFF 表处于“只读访问”和“正在装入”状态:

```
表状态:
     正在装入
     只读访问
```

装入异常表

装入异常表是在装入操作期间违反了唯一索引规则、范围限制和安全策略的所有行的组合报告。可以通过使用 LOAD 命令的 FOR EXCEPTION 子句来指定装入异常表。

限制: 异常表不能包含标识列或任何其他类型的生成列。如果主表包含标识列，那么异常表中相应的列只能包含该列的类型、长度和可空性属性。此外，异常表不能是分区表，也不能带有唯一索引。

load 实用程序使用的异常表与 SET INTEGRITY 语句使用的异常表完全相同。它是用户创建的表，反映正在装入的表的定义，并包含一些附加的列。

可以对正在装入的表所在的表空间指定装入异常表，也可以对另一个表空间指定该表。在这两种情况下，都应该对同一个数据库分区组指定装入异常表和正在装入的表，并确保这两个表使用相同的分布键。

何时使用异常表

当装入的数据包含唯一索引并且可能具有重复记录时，请使用异常表。如果未指定异常表，但却找到重复的记录，那么装入操作将继续执行，并且仅发出关于已删除的重复记录的警告消息。不会对重复的记录进行日志记录。

装入操作完成后，可以使用异常表中的信息来更正发生错误的数据。然后，可以将更正过的数据插入到表中。

行将被追加到异常表中现有信息后面。由于未进行检查，不能确保表为空，所以只将新信息添加到先前装入操作返回的无效行中。如果只需要当前装入操作返回的无效行，那么可以在调用 load 实用程序之前除去现有行。此外，在定义装入操作时，可以指定让异常表记录发现违例的时间以及所违反约束的名称。

由于会日志记录每个删除事件，所以如果有大量记录违反唯一性条件，那么在装入的删除阶段可能会添满日志。

对于在构建索引前由于数据无效而拒绝的任何行来说，不会将它们插入到异常表中。

装入失败或不完整

重新启动中断的装入操作

如果在装入操作期间出现故障或中断，那么可以使用 load 实用程序终止该操作、重新装入表或重新启动装入操作。

如果 load 实用程序因为用户错误（例如，数据文件不存在或列名无效）而不能启动，那么操作将终止并让目标表处于正常状态。

装入操作开始时，目标表将处于“正在装入”状态。出现故障时，表状态将更改为“装入暂挂”。要使表脱离该状态，可以发出 LOAD TERMINATE 以回滚操作、发出 LOAD REPLACE 以重新装入整个表，或者发出 LOAD RESTART。

通常，在这种情况下，最好重新启动装入操作。由于 load 实用程序从装入操作最后成功到达的位置而不是从该操作的开头重新启动装入操作，所以这样做可以节省时间。操作重新启动的准确位置取决于在原始命令中指定的参数。如果指定了 SAVECOUNT 选项，并且上一个装入操作在装入阶段失败，那么装入操作将在它到达的最后一个一致点重新启动。否则，装入操作在成功到达的最后一个阶段（装入、构建或删除阶段）开始时重新启动。

如果要装入 XML 文档，那么该行为稍有不同。因为在装入 XML 数据时不支持 SAVECOUNT 选项，所以在装入阶段期间失败的装入操作将要从操作的起始处重新启动。正如其他数据类型一样，如果在构建阶段期间装入失败，那么将在 REBUILD 方式下构建索引，因此会扫描该表以便从每一行获取所有索引键；但是，也必须扫描每个 XML 文档以获取索引键。扫描 XML 文档以查找索引键的这一过程要求对它们重新进行语法分析，这是成本高昂的操作。而且，诸如区域和路径索引之类的内部 XML 索引需要先重构，这也要求扫描 XDA 对象。

解决导致装入操作失败的情况后，重新发出 LOAD 命令。确保指定的参数与原始命令中的参数完全相同，以便 load 实用程序可找到必需的临时文件。如果要禁止读访问，那么不必指定完全相同的参数。还可以将指定了 ALLOW READ ACCESS 选项的装入操作作为 ALLOW NO ACCESS 选项重新启动。

注: 请不要删除或修改 load 实用程序创建的任何临时文件。

如果下列命令产生的装入操作失败，

```
LOAD FROM file_name OF file_type
SAVECOUNT n
MESSAGES message_file
load_method
INTO target_tablename
```

您可以通过将指定的装入方法（load_method）替换为 RESTART 方法来重新启动该操作:

```
LOAD FROM file_name OF file_type
SAVECOUNT n
MESSAGES message_file
RESTART
INTO target_tablename
```

不能重新启动的失败装入

如果失败或中断的装入操作中使用的表处于“不可重新启动装入”状态，那么不能重新启动该操作。因为下列原因，表将处于该状态:

- 在未成功地重新启动或终止的失败装入操作后执行前滚操作
- 根据表处于“正在装入”或“装入暂挂”状态时创建的联机备份执行复原操作

应该发出 LOAD TERMINATE 或 LOAD REPLACE 命令。

重新启动或终止 ALLOW READ ACCESS 装入操作

还可以使用 ALLOW READ ACCESS 选项重新启动或终止指定了 ALLOW READ ACCESS 选项的已中断或已取消的装入操作。使用 ALLOW READ ACCESS 选项允许其他应用程序在执行终止或重新启动操作时查询表数据。对于 ALLOW READ ACCESS 方式的装入操作，在落实数据之前，该表以独占方式锁定。

如果索引对象不可用或标记为无效，那么不允许 ALLOW READ ACCESS 方式的装入重新启动或终止操作。

如果原始装入操作在索引复制阶段被中断或取消，那么因为索引可能已损坏而不允许 ALLOW READ ACCESS 方式的重新启动操作。

如果 ALLOW READ ACCESS 方式的装入操作在装入阶段被中断或取消，那么它将在装入阶段重新启动。如果它在装入阶段以外的任何阶段被中断或取消，那么它将在构建阶段重新启动。如果原始装入操作处于 ALLOW NO ACCESS 方式，那么当原始装入操作到达该点并且索引有效时，将在删除阶段发生重新启动操作。如果索引标记为无效，那么 load 实用程序将从构建阶段重新启动装入操作。

注: 即使指定了 INDEXING MODE INCREMENTAL 选项，所有装入重新启动操作也会选择 REBUILD 建立索引方式。

发出 LOAD TERMINATE 命令通常会导致已中断或已取消的装入操作以最短延迟回滚。但是，对指定了 ALLOW READ ACCESS 和 INDEXING MODE INCREMENTAL 的装入操作发出 LOAD TERMINATE 命令时，load 实用程序扫描索引和校正任何不一致时会有延迟。此延迟的长度取决于索引的大小，并且无论是否对装入终止操作指定 ALLOW READ ACCESS 选项，都会发生延迟。如果原始装入操作在到达构建阶段前失败，那么不会发生延迟。

注: 因为校正索引中的不一致而导致的延迟明显比因为将索引标记为无效并进行重建的延迟要短。

不能对处于“不可重新启动装入”状态的表执行装入重新启动操作。在前滚操作期间，表可能处于“不可重新启动装入”状态。如果前滚至装入操作结尾之前的时间点，或者前滚完已中断或已取消的装入操作但未前滚至装入终止或装入重新启动操作的结尾，那么可能发生这种情况。

使用装入副本位置文件来恢复数据

DB2LOADREC 注册表变量用来标识包含装入副本位置信息的文件。在前滚恢复操作期间，将使用此文件来找到装入副本。

DB2LOADREC 具有下列各方面的信息:

- 介质类型
- 要使用的介质设备数
- 表装入操作期间生成的装入副本的位置
- 装入副本的文件名（如果适用的话）

如果该位置文件不存在，或者在该文件中找不到匹配的条目，那么将使用日志记录中的信息。

在执行前滚恢复操作之前，此文件中的信息可能会被覆盖。

注:

1. 在多分区数据库中，必须使用 db2set 命令为所有数据库分区服务器设置 **DB2LOADREC** 注册表变量。
2. 在多分区数据库中，每台数据库分区服务器上都必须存在装入副本文件，并且文件名（包括路径）必须相同。
3. 如果 **DB2LOADREC** 注册表变量标识的文件包含无效的条目，那么将使用旧的装入副本位置文件来提供信息以替换无效条目。

位置文件提供了以下信息。前 5 个参数值必须有效，它们用来标识装入副本。所记录的每个装入副本都重复替代一次整个结构。例如:

```
TIMestamp      19950725182542         * Time stamp generated at load time
DBPartition    0                      * DB Partition number (OPTIONAL)
SCHema         PAYROLL                * Schema of table loaded
TABlename      EMPLOYEES              * Table name
DATabasename   DBT                    * Database name
DB2instance    toronto                * DB2INSTANCE
BUFfernumber   NULL                   * Number of buffers to be used for
                                       recovery
SESsionnumber  NULL                   * Number of sessions to be used for
                                       recovery
TYPeofmedia    L                      * Type of media - L for local device
```

```
                                                        A for TSM
                                                        O for other vendors
LOCationnumber 3                      * Number of locations
   ENTry       /u/toronto/dbt.payroll.employes.001
   ENT         /u/toronto/dbt.payroll.employes.002
   ENT         /dev/rmt0
TIM            19950725192054
DBP            18
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            A
TIM            19940325192054
SCH            PAYROLL
TAB            DEPT
DAT            DBT
DB2            toronto
BUF            NULL
SES            NULL
TYP            O
SHRlib         /@sys/lib/backup_vendor.a
```

注:

1. 每个关键字的前三个字符有意义。所有关键字必须按指定的顺序出现。不接受空行。
2. 时间戳记的格式为 *yyyymmddhhmmss*。
3. 除 BUF 和 SES 字段（它们可以是 NULL）以及 DBP（可以从列表中省略它）以外，所有字段都是必需字段。如果 SES 是 NULL，那么将使用 *dft_loadrec_ses* 配置参数指定的值。如果 BUF 是 NULL，那么缺省值是 SES+2。
4. 即使位置文件只包含一个无效条目，也会使用先前装入副本位置文件来提供那些值。
5. 介质类型可以是本地设备（L 表示磁带、磁盘或软盘）、TSM（A）或其他供应商的设备（O）。如果类型为 L，那么需要位置数，后跟位置条目。如果类型为 A，那么不需要更多输入。如果类型为 O，那么需要共享库名。
6. SHRlib 参数指向能够存储装入副本数据的库。
7. 如果在指定了 COPY NO 或 NONRECOVERABLE 选项的情况下调用装入操作，并且在操作完成后未备份数据库或受影响的表空间，那么无法将数据库或表空间恢复到装入操作完成后的时间点。即，无法使用前滚恢复操作来重新创建数据库或表空间以将它们恢复到装入操作完成后所处的状态。只能将数据库或表空间恢复到执行装入操作前的时间点。

如果要使用特定的装入副本，那么可以使用数据库的恢复历史记录文件来确定该特定装入操作的时间戳记。在多分区数据库中，恢复历史记录文件位于各个数据库分区本地。

装入转储文件

通过指定 dumpfile 文件类型修饰符，可以向 load 实用程序指示异常文件的名称和位置，load 实用程序会将拒绝的行写入该文件。

在分区数据库环境中运行时，分区子代理程序或装入子代理程序可能会拒绝行。因此，对转储文件名指定的扩展名标识了子代理程序类型以及生成异常的数据库分区号。例如，如果指定了以下转储文件值:

```
dumpfile = "/u/usrname/dumpit"
```

装入子代理程序在数据库分区 5 上拒绝的行将存储在 /u/usrname/dumpit.load.005 文件中，装入子代理程序在数据库分区 2 上拒绝的行将存储在 /u/usrname/dumpit.load.002 文件中，分区子代理程序在数据库分区 2 上拒绝的行将存储在 /u/usrname/dumpit.part.002 文件中，依此类推。

对于装入子代理程序拒绝的行来说，如果行长度小于 32768 字节，那么将把整个记录复制到转储文件中；如果超过此长度，就会将行片段（包括记录的最终字节）写入该文件。

对于分区子代理程序拒绝的行来说，将把整行复制到转储文件中，而不考虑记录大小。

装入临时文件

DB2 将在装入处理期间创建临时二进制文件。这些文件用于装入崩溃恢复、装入终止操作、警告和错误消息以及运行时控制数据。

装入临时文件将在装入操作完成而未发生任何错误时除去。临时文件将写至通过 LOAD 命令的 *temp-pathname* 参数或 db2Load API 的 *piTempFilesPath* 参数指定的路径。缺省路径为数据库目录的子目录。

临时文件路径在服务器上，并且由 DB2 实例以独占方式访问。因此，对 *temp-pathname* 参数指定的任何路径名限定都必须反映服务器（而不是客户机）的目录结构，并且 DB2 实例所有者对该路径必须具有读写许可权。

注: 在 MPP 系统中，临时文件路径应该在本地磁盘上而不是在 NFS 安装上。如果该路径在 NFS 安装上，那么在装入操作期间性能会显著下降。
警告: 写至此路径的临时文件在任何情况下都不能篡改。篡改这些临时文件将导致装入操作失败，并且会使数据库陷入危险状况。

load 实用程序日志记录

该实用程序管理器产生与若干 DB2 实用程序（包括 load 实用程序）相关联的日志记录。

下列日志记录标记装入操作期间特定活动的开始或结束:

- 装入开始。此日志记录与装入操作的开始相关联。
- 装入删除开始。此日志记录与装入操作的删除阶段的开始相关联。仅当存在重复主键值时，才会开始删除阶段。在删除阶段期间，针对表记录或索引键的每个删除操作都会记录下来。
- 装入删除结束。此日志记录与装入操作的删除阶段的结尾相关联。在成功装入操作的前滚恢复期间，将重复执行此删除阶段。
- 装入暂挂列表。此日志记录将在落实装入事务时写入并且用来代替常规事务落实日志记录。

以下列表概述 load 实用程序根据输入数据的大小创建的日志记录:

- 将对 load 实用程序在 DMS 表空间中分配或删除的每个表空间扩展数据块创建两个日志记录。
- 将为使用的每个标识值块创建一个日志记录。
- 将为在装入操作的删除阶段删除的每个数据行或索引键创建日志记录。
- 将创建日志记录以在执行指定了 ALLOW READ ACCESS 和 INDEXING MODE INCREMENTAL 选项的装入操作时维护索引树的完整性。记录的记录数比完整记录的插入到索引中的记录数明显要小。

装入概述 - 分区数据库环境

在多分区数据库环境中，大量的数据放在多个数据库分区中。分布键用来确定每部分数据所在的数据库分区。必须先*分布*数据，然后才能将该数据装入到正确的数据库分区中。

在多分区数据库中装入表时，load 实用程序可以:

- 并行地分布输入数据
- 同时在各个相应数据库分区中装入数据
- 将数据从一个系统传输到另一个系统

将数据装入到多分区数据库中分两阶段完成: 第一阶段为*设置阶段*，在此阶段获取数据库分区资源（如表锁定）；第二阶段为*装入阶段*，在此阶段将数据装入到数据库分区中。可以使用 LOAD 命令的 ISOLATE_PART_ERRS 选项来选择这些阶段的错误处理方式，并可以选择一个或多个数据库分区上的错误对未发生错误的数据库分区上的装入操作的影响。

在将数据装入多分区数据库时，可以使用下列其中一种方式:

PARTITION_AND_LOAD
对数据进行分布（有可能以并行方式进行分布），并且同时在各个相应数据库分区上装入数据。

PARTITION_ONLY
对数据进行分布（有可能以并行方式进行分布），并将输出写入每个装入数据库分区上指定位置中的文件。每个文件都包含分区头，该分区头指定数据在数据库分区上的分布方式，并指定可以使用 LOAD_ONLY 方式将该文件装入到数据库中。

LOAD_ONLY
假定数据已分布在数据库分区上；将跳过分布过程，并且在相应的数据库分区上同时装入数据。

LOAD_ONLY_VERIFY_PART
假定数据已分布在数据库分区上，但数据文件未包含分区头。将跳过分布过程，并且在相应的数据库分区上同时装入数据。在装入操作期间，将检查每一行以验证它是否在正确的数据库分区中。如果指定了 `dumpfile` 文件类型修饰符，那么会将发生数据库分区违例的行放到转储文件中。否则会删除那些行。如果特定装入数据库分区上存在数据库分区违例，那么会将一条有关该数据库分区的警告写至装入消息文件。

ANALYZE
生成最佳分布图（在所有数据库分区之间均匀地分布数据）。

概念和术语

在讨论 load 实用程序在带有多个数据库分区的分区数据库环境中的行为和操作时，将使用以下术语:

- *协调程序分区*是一个数据库分区，用户连接到该分区以执行装入操作。在 PARTITION_AND_LOAD、PARTITION_ONLY 和 ANALYZE 方式下，除非指定了 LOAD 命令的 CLIENT 选项，否则假定数据文件在此数据库分区上。如果指定 CLIENT，那么表示要装入的数据在连接的远程客户机上。
- 在 PARTITION_AND_LOAD、PARTITION_ONLY 和 ANALYZE 方式下，*预分区代理程序*读取用户数据并以循环方式将其分发给*分区代理程序*，后者将分布该数据。此过程始终是在协调程序分区上执行的。对于任何装入操作，每个数据库分区最多允许一个分区代理程序。
- 在 PARTITION_AND_LOAD、LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式下，在每个输出数据库分区上都运行*装入代理程序*，它协调该数据库分区上的数据装入操作。
- 在 PARTITION_ONLY 装入操作期间，在每个输出数据库分区上运行*"装入到文件"代理程序*。它们从分区代理程序接收数据并将该数据写入所在数据库分区上的文件中。
- SOURCEUSEREXIT 选项提供了一种工具，load 实用程序可通过该工具执行定制脚本或可执行文件（此处称为*用户出口*）。

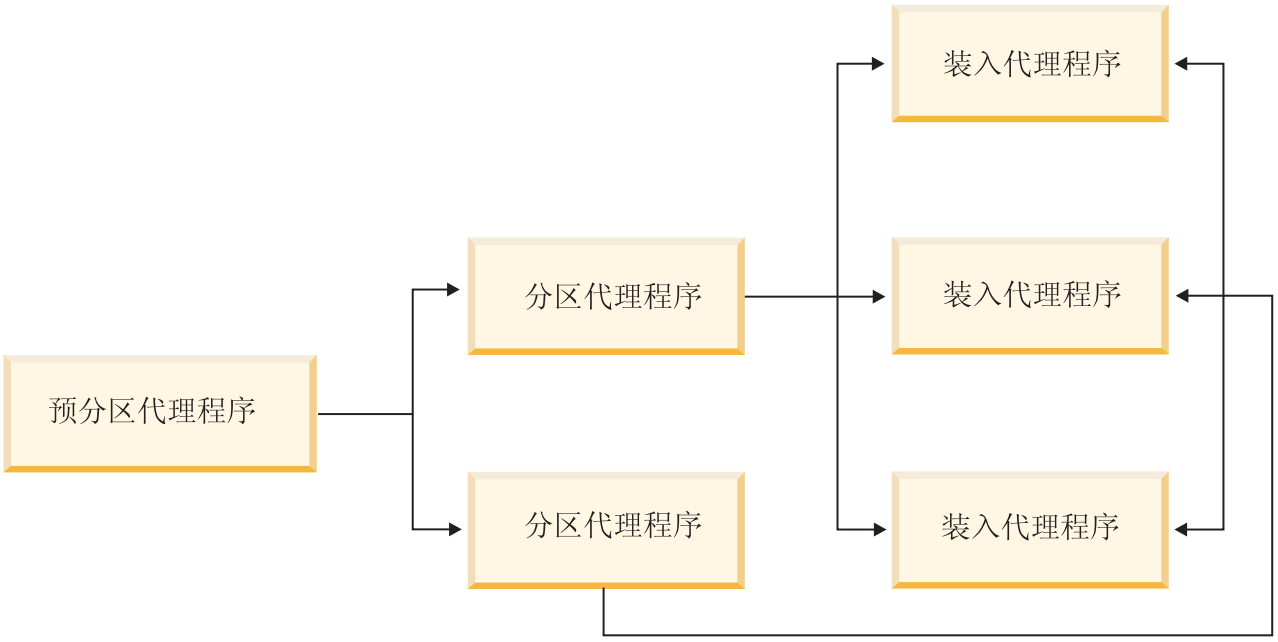


图 13. 分区数据库装入概述. 预分区代理程序读取源数据，然后向两个分区代理程序中的每个分区代理程序各发送接近一半的数据，这两个分区代理程序分布数据并将该数据发送给三个数据库分区中的一个数据库分区。每个数据库分区上的装入代理程序装入数据。

在分区数据库环境中装入数据

使用 load 实用程序将数据装入到分区数据库环境中。

在多分区数据库中装入表之前:

1. 确保正确设置了 *svcename* 数据库管理器配置参数和 **DB2COMM** 概要文件注册表变量。由于 load 实用程序使用 TCP/IP 来将数据从预分区代理程序传输至分区代理程序以及从分区代理程序传输至装入数据库分区，所以正确进行上述设置非常重要。
2. 在调用 load 实用程序前，您必须连接至（或者能够隐式地连接至）要装入数据的数据库。由于 load 实用程序将发出 COMMIT 语句，所以，在开始执行装入操作前，应该通过发出 COMMIT 或 ROLLBACK 语句完成所有事务并释放任何锁定。如果使用的是 PARTITION_AND_LOAD、PARTITION_ONLY 或 ANALYZE 方式，那么装入的数据文件必须在此数据库分区上，但在下列情况下除外:
   a. 指定了 CLIENT 选项，在这种情况下，数据必须在客户机上;
   b. 输入源类型为 CURSOR，在这种情况下，没有输入文件。
3. 运行设计顾问程序以确定每个表的最佳数据库分区。有关更多信息，请参阅《*调整数据库性能*》中的『设计顾问程序』。

在使用 load 实用程序以在多分区数据库环境中装入数据时，下列限制适用:

- 装入操作的输入文件位置不能是磁带设备。
- 除非使用 ANALYZE 方式，否则不支持 ROWCOUNT 选项。
- 如果目标表带有进行分布所需的标识列，并且未指定 `identityoverride` 文件类型修饰符，或者如果正在使用多个数据库分区来分布数据并接着装入该数据，那么不支持在 LOAD 命令上使用大于 0 的 SAVECOUNT。
- 如果分布键包含标识列，那么只支持 PARTITION_AND_LOAD 方式。
- LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式不能与 LOAD 命令的 CLIENT 选项配合使用。
- LOAD_ONLY_VERIFY_PART 方式不能与 CURSOR 输入源类型配合使用。
- 分布错误隔离方式 LOAD_ERRS_ONLY 和 SETUP_AND_LOAD_ERRS 不能与 LOAD 命令的 ALLOW READ ACCESS 和 COPY YES 选项配合使用。
- 如果 OUTPUT_DBPARTNUMS 和 PARTITIONING_DBPARTNUMS 选项指定的数据库分区不重叠，那么多个装入操作可以同时将数据装入到同一个表中。例如，如果表是在数据库分区 0 至 3 上定义的，那么一个装入操作可以将数据装入到数据库分区 0 和 1 中，而另一个装入操作可以将数据装入到数据库分区 2 和 3 中。
- 对于跨多个数据库分区的表来说，只能将非定界 ASCII（ASC）和定界 ASCII（DEL）文件分布到这些表中。不能分布 PC/IXF 文件，但可使用 LOAD_ONLY_VERIFY_PART 方式的装入操作将 PC/IXF 文件装入到分布在多个数据库分区中的表中。

下列示例说明如何使用 LOAD 命令来启动各种类型的装入操作。下列示例中使用的数据库有 5 个数据库分区: 0、1、2、3 和 4。每个数据库分区都有一个本地目录 `/db2/data/`。在数据库分区 0、1、3 和 4 上定义了两个表 TABLE1 和 TABLE2。从客户机装入数据时，用户能够访问除数据库分区以外的远程客户机。

从服务器分区中装入

分布和装入示例

在此示例中，您连接到一个数据库分区，该数据库分区可能是也可能不是用来定义 TABLE1 的数据库分区。数据文件 load.del 在此数据库分区的当前工作目录中。要将 load.del 中的数据装入到所有定义了 TABLE1 的数据库分区中，请发出以下命令:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

注: 在此示例中，将对分区数据库环境的所有配置参数使用缺省值: MODE 参数将缺省为 PARTITION_AND_LOAD，OUTPUT_DBPARTNUMS 选项将缺省为所有定义 TABLE1 的数据库分区，而 PARTITIONING_DBPARTNUMS 将缺省为在未指定数据库分区时根据 LOAD 命令规则选择的一组数据库分区。

要在数据分布在数据库分区 3 和数据库分区 4 上的位置执行装入操作，请发出以下命令:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                    PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

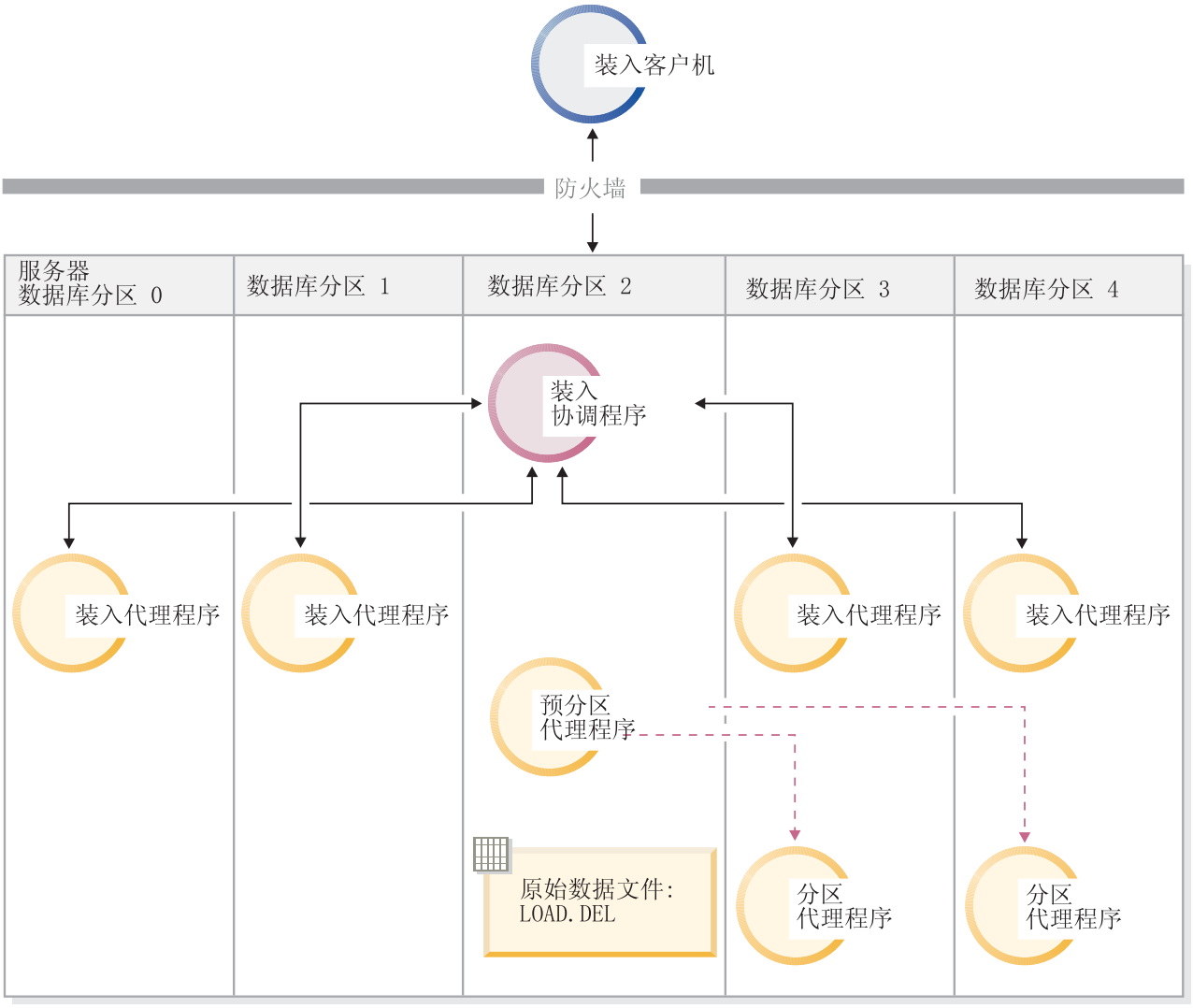


图 14. . 此图说明发出以上命令后产生的行为。数据将装入到数据库分区 3 和 4 中。

仅分布示例

在此示例中，您连接到一个数据库分区，该数据库分区可能是也可能不是用来定义 TABLE1 的数据库分区。数据文件 load.del 在此数据库分区的当前工作目录中。在使用数据库分区 3 和数据库分区 4 的情况下，要将 load.del 分布（而不装入）到所有定义 TABLE1 的数据库分区中，请发出以下命令:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                               PART_FILE_LOCATION /db2/data
                                               PARTITIONING_DBPARTNUMS (3,4)
```

这将导致把文件 load.del.xxx 存储在每个数据库分区上的 /db2/data 目录中，其中 xxx 是 3 位的数据库分区号。

在仅使用数据库分区 0（缺省 PARTITIONING_DBPARTNUMS 值）上运行的 1 个分区代理程序的情况下，要将 load.del 文件分布到数据库分区 1 和 3，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                               PART_FILE_LOCATION /db2/data
                                               OUTPUT_DBPARTNUMS (1,3)
```

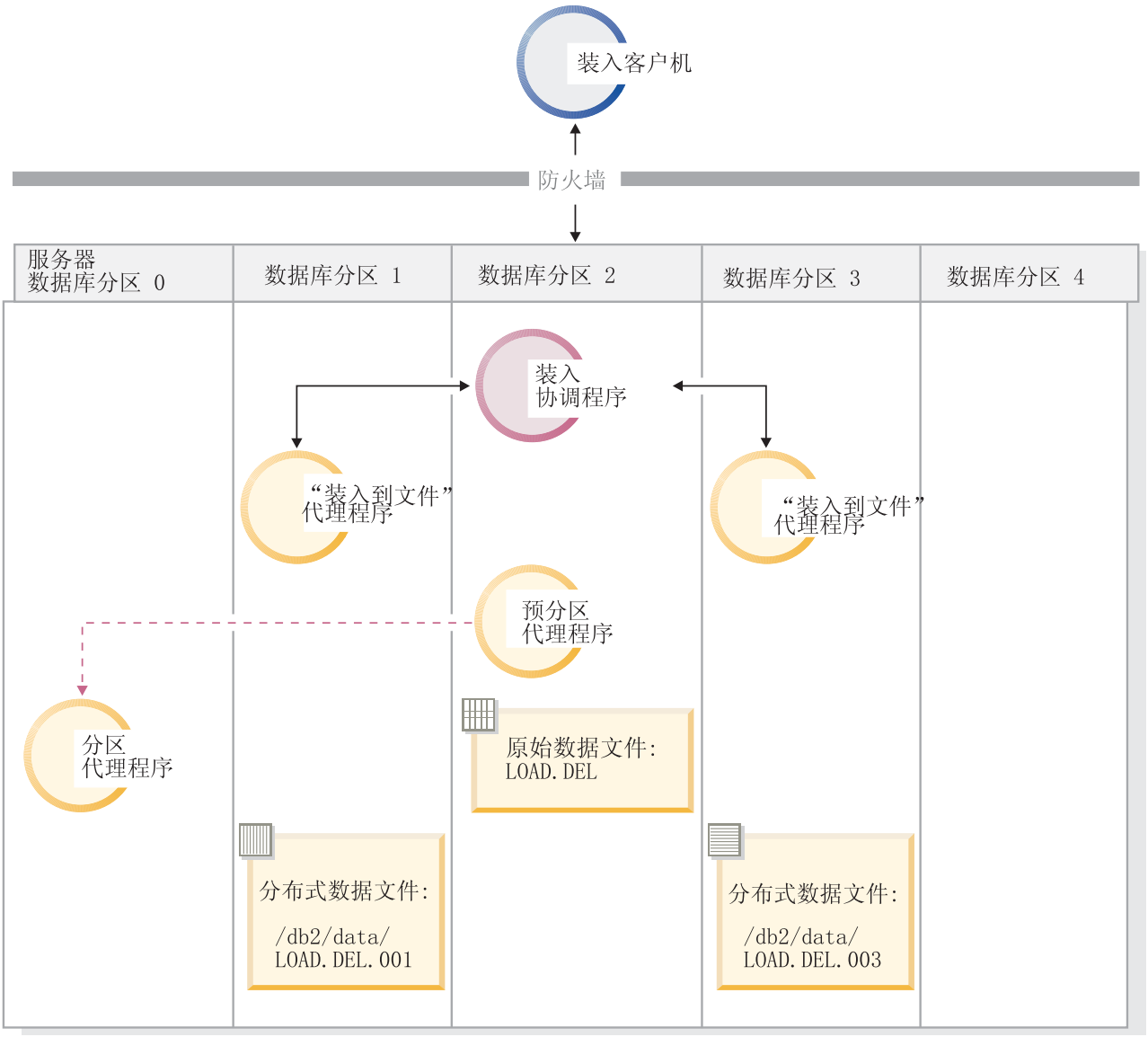


图 15. . 此图说明发出以上命令后产生的行为。将使用数据库分区 0 上运行的 1 个分区代理程序将数据装入到数据库分区 1 和 3 中。

仅装入示例

如果已经以 PARTITION_ONLY 方式执行了装入操作，并且要将每个装入数据库分区的 /db2/data 目录中的分区文件装入到所有定义了 TABLE1 的数据库分区中，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                               PART_FILE_LOCATION /db2/data
```

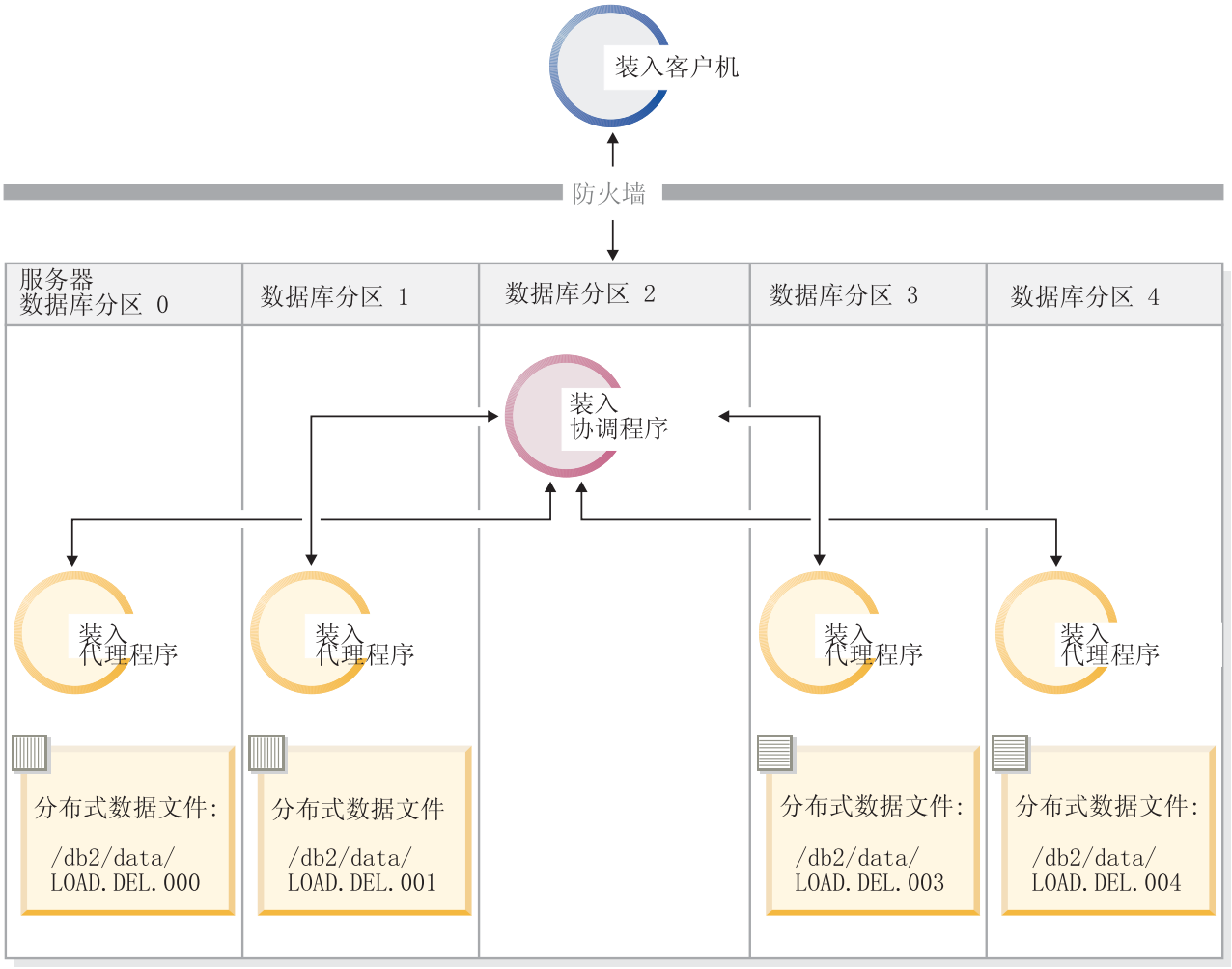


图 16. . 此图说明发出以上命令后产生的行为。将把分布式数据装入到所有用来定义 TABLE1 的数据库分区中。

要仅装入到数据库分区 4 中，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                              PART_FILE_LOCATION /db2/data
                                              OUTPUT_DBPARTNUMS (4)
```

装入不带分布图头的预分布文件

可以使用 LOAD 命令来将不带分布头的装入数据文件直接装入到数个数据库分区中。如果数据文件在每个用来定义 TABLE1 的数据库分区上的 /db2/data 目录中，并且名为 load.del.xxx（其中 xxx 是数据库分区号），那么可以通过发出以下命令来装入那些文件:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                        REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                              PART_FILE_LOCATION /db2/data
```

要仅将数据装入到数据库分区 1 中，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                         REPLACE INTO TABLE1
                         PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                                    PART_FILE_LOCATION /db2/data
                                                    OUTPUT_DBPARTNUMS (1)
```

注: 如果指定了转储文件，那么将拒绝装入不属于源数据库分区的行并将它们放入转储文件。

从远程客户机装入至多分区数据库

要将远程客户机上的文件中的数据装入到多分区数据库中，必须指定 LOAD 命令的 CLIENT 选项以指示数据文件不在服务器分区上。例如:

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

注: 不能将 LOAD_ONLY 或 LOAD_ONLY_VERIFY_PART 方式与 CLIENT 选项配合使用。

从游标装入

与在单一分区数据库中一样，可以从游标装入到多分区数据库中。在此示例中，对于 PARTITION_ONLY 和 LOAD_ONLY 方式，PART_FILE_LOCATION 选项必须指定标准文件名。此名称是在每个输出数据库分区上创建或装入的分布文件的标准基本文件名。如果目标表包含 LOB 列，那么可以使用指定的基本名称来创建多个文件。

要将语句 `SELECT * FROM TABLE1` 的应答集中的所有行分布至名为 `/db2/data/select.out.xxx`（其中 xxx 是数据库分区号）的每个数据库分区上的文件，以便将来装入到 TABLE2 中，请发出以下命令:

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                         PARTITIONED DB CONFIG MODE PARTITION_ONLY
                         PART_FILE_LOCATION /db2/data/select.out
```

然后，可以通过发出以下 LOAD 命令来装入以上操作生成的数据文件:

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                         PARTITIONED CB CONFIG MODE LOAD_ONLY
                         PART_FILE_LOCATION /db2/data/select.out
```

在分区数据库环境中装入数据 - 提示与技巧

以下是在多分区数据库中装入表前要考虑的一些信息:

- 对少量数据使用 load 实用程序，熟悉装入配置选项。
- 如果输入数据已排序或者具有某种选择的顺序，并且要在装入过程中维护该顺序，那么只应该将一个数据库分区用于分布。并行分布无法保证按照数据的接收顺序来装入该数据。缺省情况下，如果在 LOAD 命令中未指定 `anyorder` 修饰符，那么 load 实用程序就会选择单分区代理程序。
- 如果正在从不同的文件装入大对象（LOB）（即，如果使用 load 实用程序时指定了 `lobsinfile` 修饰符），那么所有执行装入的数据库分区都必须能够对所有包含 LOB 文件的目录进行读访问。处理 LOB 时，LOAD *lob-path* 参数必须是标准路径。

- 通过将 ISOLATE_PART_ERRS 选项设置为 SETUP_ERRS_ONLY 或 SETUP_AND_LOAD_ERRS，可以强制在多分区数据库中运行的作业继续运行，即使装入操作在启动时检测到某些装入数据库分区或相关表空间或表处于脱机状态亦如此。
- 使用 STATUS_INTERVAL 装入配置选项来监视在多分区数据库中运行的作业的进度。装入操作按指定的时间间隔生成消息，以指示预分区代理程序已读取的数据的兆字节数。这些消息将被转储到预分区代理程序消息文件中。要在装入操作期间查看此文件的内容，请连接到协调程序分区并对目标表发出 LOAD QUERY 命令。
- 如果参与分布过程的数据库分区（由 PARTITIONING_DBPARTNUMS 选项定义）与装入数据库分区（由 OUTPUT_DBPARTNUMS 选项定义）不同，就会由于 CPU 周期争用情况减少而提高性能。将数据装入到多分区数据库中时，对未参与分布或装入操作的数据库分区调用 load 实用程序。
- 如果在 LOAD 命令中指定了 MESSAGES 参数，就会将预分区代理程序、分区代理程序和装入代理程序生成的消息文件保存下来，以供装入操作完成后参考。要在装入操作期间查看这些文件的内容，请连接到期望的数据库分区并对目标表发出 LOAD QUERY 命令。
- load 实用程序仅选择一个输出数据库分区以便在该分区中收集统计信息。可以使用 RUN_STAT_DBPARTNUM 数据库配置选项来指定该数据库分区。
- 在多分区数据库中装入数据之前，请运行设计顾问程序以确定每个表的最佳分区。有关更多信息，请参阅《*调整数据库性能*》中的『设计顾问程序』。

故障诊断

如果 load 实用程序挂起，您可以:

- 使用 STATUS_INTERVAL 参数来监视多分区数据库装入操作的进度。将把状态时间间隔信息转储到协调程序分区上的预分区代理程序消息文件中。
- 检查分区代理程序消息文件，了解每个数据库分区上的分区代理进程状态。如果执行装入操作时未出错，并且设置了 TRACE 选项，那么这些消息文件应该会包含许多记录的跟踪消息。
- 检查装入消息文件以了解是否有任何装入错误消息。

  注: 必须指定 LOAD 命令的 MESSAGES 选项，这样这些文件才能存在。
- 如果有错误指示某个装入进程出错，请中断当前装入操作。

使用 LOAD QUERY 命令在分区数据库环境中监视装入操作

在分区数据库环境中执行装入操作期间，某些装入进程会在它们执行时所在的数据库分区上创建消息文件。

这些消息文件存储装入执行期间生成的所有参考消息、警告消息和错误消息。用户可以查看以下三个能生成的消息文件的装入进程: 装入代理程序、预分区代理程序和分区代理程序。只有在装入操作完成后，消息文件的内容才可用。

在装入操作期间，可连接至各个数据库分区并对目标表发出 LOAD QUERY 命令。从 CLP 中发出此命令时，此命令将显示由 LOAD QUERY 命令指定的表的数据库分区上当前存在的所有消息文件的内容。

例如，在数据库 WSDB 中，表 TABLE1 是在数据库分区 0 至 3 上定义的。您将连接至数据库分区 0 并发出以下 LOAD 命令:

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

此命令将启动装入操作，该操作包括：在数据库分区 0、1、2 和 3 上运行的装入代理程序；在数据库分区 1 上运行的分区代理程序；在数据库分区 0 上运行的预分区代理程序。

数据库分区 0 将包含预分区代理程序的消息文件以及该数据库分区上装入代理程序的消息文件。要同时查看这些文件的内容，请启动新会话并从 CLP 中发出下列命令:

```
set client connect_node 0
connect to wsdb
load query table table1
```

数据库分区 1 将包含装入代理程序的消息文件和分区代理程序的消息文件。要查看这些文件的内容，请启动新会话并从 CLP 中发出下列命令:

```
set client connect_node 1
connect to wsdb
load query table table1
```

注: STATUS_INTERVAL 装入配置选项生成的消息将出现在预分区代理程序消息文件中。要在装入操作期间查看这些消息，必须连接至协调程序分区并发出 LOAD QUERY 命令。

保存消息文件内容

如果通过 db2Load API 启动装入操作，那么必须指定消息选项（piLocalMsgFileName），将消息文件从服务器传输到客户机并存储下来以供查看。

对于从 CLP 启动的多分区数据库装入操作来说，不会在控制台上显示或保留消息文件。要在多分区数据库装入完成后保存或查看这些文件的内容，必须指定 LOAD 命令的 MESSAGES 选项。如果使用了此选项，那么装入操作一旦完成，就会将每个数据库分区上的消息文件传输到客户机并使用 MESSAGES 选项中指示的基本名称存储在文件中。对于多分区数据库装入操作，下面列示了各个装入进程所生成的消息文件名:

进程类型	文件名
装入代理程序	<message-file-name>.LOAD.<dbpartition-number>
分区代理程序	<message-file-name>.PART.<dbpartition-number>
预分区代理程序	<message-file-name>.PREP.<dbpartition-number>

例如，如果 MESSAGES 选项指定 /wsdb/messages/load，那么数据库分区 2 的装入代理程序消息文件将是 /wsdb/messages/load.LOAD.002。

注: 强烈建议从 CLP 中启动的多分区数据库装入操作使用 MESSAGES 选项。

在分区数据库环境中继续、重新启动或终止装入操作

如果装入操作在分区数据库环境中失败，那么接下来需要执行的操作取决于出现故障的时间。

多分区数据库中的装入过程由两个阶段组成:

1. *设置阶段*，在该阶段中获取数据库分区级别的资源，例如，输出数据库分区上的表锁定

   通常，如果设置阶段发生故障，不必执行重新启动和终止操作。您需要执行的操作取决于对失败的装入操作指定的错误隔离方式。

   如果装入操作指定不隔离设置阶段错误，那么整个装入操作将被取消并且每个数据库分区上的表状态都将回滚到该表在装入操作前所处的状态。

   如果装入操作指定要隔离设置阶段错误，那么装入操作将在成功完成设置阶段的数据库分区上继续，但每个失败数据库分区上的表都将回滚到该表在装入操作前所处的状态。这意味着如果某些分区在设置阶段失败，而其他分区在装入阶段失败，那么单个装入操作可能在不同阶段失败。
2. *装入阶段*，在该阶段中格式化数据并将它们装入到数据库分区上的表中。

   在多分区数据库装入操作的装入阶段，如果装入操作在至少一个数据库分区上失败，那么必须发出 load RESTART 或 load TERMINATE 命令。因为多分区数据库中的数据装入操作将通过单个事务完成，所以有必要执行此操作。

   如果您可以解决导致装入操作失败的问题，那么应该选择 load RESTART。这样可以节省时间，因为在启动了装入重新启动操作时，装入操作会在所有数据库分区上的中断位置继续执行。

   如果您希望表返回到该表在初始装入操作前所处的状态，那么应该选择 load TERMINATE。

过程:

确定装入失败的时间

如果装入操作在分区环境中失败，那么您需要做的第一件事是确定操作在哪个分区上失败以及每个操作在哪个阶段失败。这可通过查看分区总结来完成。如果从 CLP 发出 LOAD 命令，那么分区总结将显示在每个 LOAD 命令的末尾（请参阅下面的示例）。如果从 db2Load API 发出 LOAD 命令，那么分区总结包含在 db2PartLoadOut 结构的 poAgentInfoList 字段中。

如果对于给定分区，“代理程序类型”有一个“LOAD”条目，那么该分区已到达装入阶段，否则在设置阶段出现故障。SQL 代码为负数表示失败。在以下示例中，装入操作在装入阶段在分区 1 上失败。

```
代理程序类型      节点      SQL 代码      结果
------------------------------------------------------------
LOAD              000       +00000000     Success.
------------------------------------------------------------
LOAD              001       -00000289     Error. May require RESTART.
------------------------------------------------------------
LOAD              002       +00000000     Success.
------------------------------------------------------------
```

```
      LOAD           003      +00000000    Success.
.
.
.
```

继续、重新启动或终止失败的装入操作

在设置阶段，只有使用值为 SETUP_ERRS_ONLY 或 SETUP_AND_LOAD_ERRS 的 ISOLATE_PART_ERRS 选项的装入应失败。对于在此阶段在至少一个输出数据库分区上失败的装入来说，可以发出 LOAD REPLACE 或 LOAD INSERT 命令。使用 OUTPUT_DBPARTNUMS 选项来仅指定装入操作在其上失败的那些数据库分区。

对于在装入阶段在至少一个输出数据库分区上失败的装入来说，发出 load RESTART 或 load TERMINATE 命令。

对于在设置阶段在至少一个输出数据库分区上失败并且在装入阶段在至少一个输出数据库分区上失败的装入来说，需要执行两个装入操作以继续失败的装入 – 一个装入操作用于设置阶段故障，另一个用于装入阶段故障，如上所述。要有效地撤销此类失败的装入操作，发出 load TERMINATE 命令。但是，在发出该命令后，您必须说明所有分区，因为没有对在设置阶段失败的分区上的表进行任何更改，并且已撤销在装入阶段失败的分区的所有更改。

例如，在数据库 WSDB 中，TABLE1 是在数据库分区 0 至 3 上定义的。发出以下命令:

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

在设置阶段，输出数据库分区 1 上出现故障。由于隔离了设置阶段错误，所以装入操作继续，但在装入阶段在分区 3 上出现故障。要继续装入操作，应发出下列命令:

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

注: 对于装入重新启动操作，将使用在 LOAD RESTART 命令中指定的选项，因此，这些选项应该与原始 LOAD 命令中指定的选项完全相同，这一点非常重要。

迁移和版本兼容性

在多分区数据库中，**DB2_PARTITIONEDLOAD_DEFAULT** 注册表变量可用来还原为 DB2 通用数据库版本 8 之前的装入行为。

注: 从版本 9.5 起，就不推荐使用 DB2_PARTITIONEDLOAD_DEFAULT 注册表变量并且在以后的发行版中可能会将其除去。

通过在多分区数据库中还原为 LOAD 命令的 DB2 UDB版本 8 之前的行为，可以将带有有效分布头的文件装入到单一数据库分区中，而不必指定任何其他分区数据库配置选项。将 **DB2_PARTITIONEDLOAD_DEFAULT** 的值设置为 NO 可以达到此目的。如果要避免修改对单一数据库分区发出 LOAD 命令的现有脚本，那么可以选择使用此选项。例如，要将分布文件装入到一个表的数据库分区 3 中（而该表所在的数据库分区组包含 4 个数据库分区），请发出以下命令:

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

然后，从 DB2 命令行处理器中发出下列命令:

```
CONNECT RESET

SET CLIENT CONNECT_NODE 3

CONNECT TO DB MYDB

LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

在多分区数据库中，当未指定多分区数据库装入配置选项时，将在所有用来定义该表的数据库分区上执行装入操作。输入文件不需要分布头，并且 MODE 选项缺省为 PARTITION_AND_LOAD。要装入单一数据库分区，必须指定 OUTPUT_DBPARTNUMS 选项。

参考 - 在分区环境中装入

分区数据库环境中的装入会话 - CLP 示例

下列示例说明如何在多分区数据库中装入数据。

数据库有四个数据库分区，其编号从 0 到 3。数据库 WSDB 是在所有数据库分区上定义的，表 TABLE1 在缺省数据库分区组中，该数据库分区组也是在所有数据库分区上定义的。

示例 1

要将用户数据文件 load.del 中的数据装入到 TABLE1 中（该文件在数据库分区 0 上），请连接到数据库分区 0，然后发出以下命令:

```
load from load.del of del replace into table1
```

如果装入操作成功，那么输出将如下所示:

```
   代理程序类型      节点      SQL 代码      结果
_____________________________________________________________
   LOAD           000       +00000000    Success.
_____________________________________________________________
   LOAD           001       +00000000    Success.
_____________________________________________________________
   LOAD           002       +00000000    Success.
_____________________________________________________________
   LOAD           003       +00000000    Success.
_____________________________________________________________
PARTITION      001       +00000000    Success.
_____________________________________________________________
PRE_PARTITION  000       +00000000    Success.
_____________________________________________________________
   结果:          成功完成了 4 个装入。
_____________________________________________________________

   分区代理程序的总结:
   读取的行数                    = 100000
   拒绝的行数                    = 0
   分区的行数                    = 100000

   装入代理程序的总结:
   读取的行数                    = 100000
   跳过的行数          = 0
   装入的行数                    = 100000
   拒绝的行数          = 0
   删除的行数          = 0
   落实的行数                    = 100000
```

输出指示在每个数据库分区上有一个装入代理程序，并且每个装入代理程序都运行成功。输出还显示在协调程序分区上运行了一个预分区代理程序，在数据库分区 1 上运行了一个分区代理程序。这些进程都成功完成并返回正常 SQL 返回码 0。统计总结显示预分区代理程序读取了 100,000 行，分区代理程序分布了 100,000 行，装入代理程序装入的总行数为 100,000。

示例 2

在以下示例中，以 PARTITION_ONLY 方式将数据装入到 TABLE1 中。分布式输出文件存储在每个输出数据库分区上的 /db/data 目录中:

```
load from load.del of del replace into table1 partitioned db config mode
partition_only part_file_location /db/data
```

以上 LOAD 命令的输出如下所示:

```
   代理程序类型      节点      SQL 代码      结果
_____________________________________________________________
LOAD_TO_FILE   000       +00000000    Success.
_____________________________________________________________
LOAD_TO_FILE   001       +00000000    Success.
_____________________________________________________________
LOAD_TO_FILE   002       +00000000    Success.
_____________________________________________________________
LOAD_TO_FILE   003       +00000000    Success.
_____________________________________________________________
PARTITION      001       +00000000    Success.
_____________________________________________________________
PRE_PARTITION  000       +00000000    Success.
_____________________________________________________________

   分区代理程序的总结:
   读取的行数                 = 100000
   拒绝的行数                 = 0
   分区的行数                 = 100000
```

输出指示在每个输出数据库分区上都运行了“装入到文件”代理程序，这些代理程序运行成功。在协调程序分区上运行了一个预分区代理程序，在数据库分区 1 上运行了一个分区代理程序。统计总结显示预分区代理程序成功读取了 100,000 行，分区代理程序成功分布了 100,000 行。由于未将任何行装入到表中，因此未显示已装入行数总结。

示例 3

要装入在以上 PARTITION_ONLY 装入操作期间生成的文件，请发出以下命令:

```
load from load.del of del replace into table1 partitioned db config mode
 load_only part_file_location /db/data
```

load 命令的输出如下所示:

```
   代理程序类型      节点      SQL 代码      结果
_____________________________________________________________
   LOAD           000       +00000000    Success.
_____________________________________________________________
   LOAD           001       +00000000    Success.
_____________________________________________________________
   LOAD           002       +00000000    Success.
_____________________________________________________________
   LOAD           003       +00000000    Success.
_____________________________________________________________
   结果:          成功完成了 4 个装入。
_____________________________________________________________
```

```
装入代理程序的总结:
读取的行数                  = 100000
跳过的行数        = 0
装入的行数                  = 100000
拒绝的行数        = 0
删除的行数        = 0
落实的行数                  = 100000
```

此输出显示每个输出数据库分区上的装入代理程序都运行成功，并且所有装入代理程序装入的总行数为 100,000。由于未执行分布操作，因此未显示分布行数总结。

示例 4 - 失败的装入操作

如果发出以下 LOAD 命令:

```
load from load.del of del replace into table1
```

并且其中一个装入数据库分区在装入操作期间耗尽表空间，那么将返回以下输出:

```
SQL0289N  不能在表空间"DMS4KT"中分配新页。
SQLSTATE=57011

代理程序类型      节点      SQL 代码      结果
------------------------------------------------------------------------
LOAD            000       +00000000     Success.
------------------------------------------------------------------------
LOAD            001       -00000289     Error. May require RESTART.
------------------------------------------------------------------------
LOAD            002       +00000000     Success.
------------------------------------------------------------------------
LOAD            003       +00000000     Success.
------------------------------------------------------------------------
PARTITION       001       +00000000     Success.
------------------------------------------------------------------------
PRE_PARTITION   000       +00000000     Success.
------------------------------------------------------------------------
结果:          成功完成了 4 个装入中的 3 个。
------------------------------------------------------------------------

分区代理程序的总结:
读取的行数                  = 0
拒绝的行数                  = 0
分区的行数                  = 0

装入代理程序的总结:
读取的行数                  = 0
跳过的行数        = 0
已装入的行数                  = 0
拒绝的行数        = 0
删除的行数        = 0
已落实的行数                  = 0
```

输出指示装入操作返回了错误 SQL0289。数据库分区总结指示数据库分区 1 耗尽空间。如果对数据库分区 1 上的表空间容器添加了更多空间，那么可以按如下方式重新启动装入操作:

```
load from load.del of del restart into table1
```

为分区数据库环境装入配置选项

MODE X

指定装入多分区数据库时装入操作采用的方式。PARTITION_AND_LOAD 是缺省值。有效值为:

- PARTITION_AND_LOAD。对数据进行分布（有可能以并行方式进行分布），并且同时在各个相应数据库分区上装入数据。
- PARTITION_ONLY。对数据进行分布（有可能以并行方式进行分布），并将输出写入每个装入数据库分区上指定位置中的文件。对于 CURSOR 以外的文件类型来说，每个数据库分区上的输出文件名格式都是 filename.xxx，其中 filename 是 LOAD 命令中指定的输入文件名，xxx 是 3 位的数据库分区号。对于 CURSOR 文件类型来说，每个数据库分区上的输出文件名由 PART_FILE_LOCATION 选项确定。请参阅 PART_FILE_LOCATION 选项以了解有关如何指定每个数据库分区的分布文件位置的详细信息。

  注:

  1. 此方式不能用于 CLI 装入操作。
  2. 如果表包含进行分布时所需的标识列，那么除非指定了 identityoverride 文件类型修饰符，否则不支持此方式。
  3. 为文件类型 CURSOR 生成的分布文件在 DB2 发行版之间不兼容。这意味着不能使用 LOAD_ONLY 方式装入在先前发行版中生成的文件类型为 CURSOR 的分布文件。同样，不能使用 LOAD_ONLY 方式在将来发行版中装入在当前发行版中生成的文件类型为 CURSOR 的分布文件。
- LOAD_ONLY。假定已对数据进行分布；将跳过分布过程，并且在相应的数据库分区上同时装入数据。对于 CURSOR 以外的文件类型来说，每个数据库分区的输入文件名格式都应该是 filename.xxx，其中 filename 是 LOAD 命令中指定的文件名，xxx 是 3 位的数据库分区号。对于 CURSOR 文件类型来说，每个数据库分区上的输入文件名由 PART_FILE_LOCATION 选项确定。请参阅 PART_FILE_LOCATION 选项以了解有关如何指定每个数据库分区的分布文件位置的详细信息。

  注:

  1. 此方式不能用于 CLI 装入操作；或者当指定了 LOAD 命令的 CLIENT 选项时，也不能使用此方式。
  2. 如果表包含进行分布时所需的标识列，那么除非指定了 identityoverride 文件类型修饰符，否则不支持此方式。
- LOAD_ONLY_VERIFY_PART。假定已对数据进行分布，但数据文件未包含分区头。将跳过分布过程，并且在相应的数据库分区上同时装入数据。在装入操作期间，将检查每一行以验证它是否在正确的数据库分区中。如果指定了 dumpfile 文件类型修饰符，就会将发生数据库分区违例的行放到转储文件中。否则会删除那些行。如果特定装入数据库分区上存在数据库分区违例，那么会将一条有关该数据库分区的警告写至装入消息文件。每个数据库分区的输入文件名格式都应该是 filename.xxx，其中 filename 是 LOAD 命令中指定的文件名，xxx 是 3 位的数据库分区号。请参阅 PART_FILE_LOCATION 选项以了解有关如何指定每个数据库分区的分布文件位置的详细信息。

  注:

  1. 此方式不能用于 CLI 装入操作；或者当指定了 LOAD 命令的 CLIENT 选项时，也不能使用此方式。
  2. 如果表包含进行分布时所需的标识列，那么除非指定了 identityoverride 文件类型修饰符，否则不支持此方式。

- ANALYZE。生成最佳分布图（在所有数据库分区之间均匀地分布数据）。

PART_FILE_LOCATION X

在 PARTITION_ONLY、LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式下，此参数可用来指定分布文件的位置。在 OUTPUT_DBPARTNUMS 选项指定的每个数据库分区上，此位置必须存在。如果指定的位置是相对路径名，那么会将该路径追加至当前目录以创建分布式文件位置。

对于 CURSOR 文件类型来说，必须指定此选项，并且位置必须引用标准文件名。在 PARTITION_ONLY 方式下，此名称是在每个输出数据库分区上创建的分布式文件的标准基本文件名，或者在 LOAD_ONLY 方式，此名称是对于每个数据库分区要读取的文件的位置。使用 PARTITION_ONLY 方式时，如果目标表包含 LOB 列，那么可以使用指定的基本名称来创建多个文件。

对于 CURSOR 以外的文件类型来说，如果未指定此选项，那么将使用当前目录来存储分布式文件。

OUTPUT_DBPARTNUMS X

X 表示数据库分区号列表。数据库分区号表示要执行装入操作的数据库分区。数据库分区号必须是定义了该表的数据库分区的子集。缺省情况下，选择了所有数据库分区。必须将此列表括在圆括号中，并且列表项必须由逗号分隔。允许指定范围（例如，(0, 2 to 10, 15)）。

PARTITIONING_DBPARTNUMS X

X 表示分布过程中使用的数据库分区号列表。必须将此列表括在圆括号中，并且列表项必须由逗号分隔。允许指定范围（例如，(0, 2 to 10, 15)）。对分布过程指定的数据库分区可能与要装入的数据库分区不同。如果未指定 PARTITIONING_DBPARTNUMS，那么 load 实用程序会确定需要的数据库分区数以及为获得最优性能而需要使用的数据库分区。

如果在 LOAD 命令中未指定 anyorder 文件类型修饰符，那么在装入会话中将只使用一个分区代理程序。此外，如果仅对 OUTPUT_DBPARTNUMS 选项指定了一个数据库分区，或者装入操作的协调程序分区不是 OUTPUT_DBPARTNUMS 的元素，那么会在分布过程中使用装入操作的协调程序分区。否则，在分布过程中使用 OUTPUT_DBPARTNUMS 中的第一个数据库分区（不是协调程序分区）。

如果指定了 anyorder 文件类型修饰符，那么按以下方式确定分布过程中使用的数据库分区数: (OUTPUT_DBPARTNUMS 中的分区数/4 + 1)。

MAX_NUM_PART_AGENTS X

指定装入会话中要使用的最大分区代理程序数。缺省值为 25。

ISOLATE_PART_ERRS X

指示装入操作如何对各个数据库分区上发生的错误作出反应。除非同时指定了 LOAD 命令的 ALLOW READ ACCESS 和 COPY YES 选项（在这种情况下，缺省值为 NO_ISOLATION），否则缺省值为 LOAD_ERRS_ONLY。有效值为:

- SETUP_ERRS_ONLY。设置期间在数据库分区上发生的错误（例如，访问数据库分区时发生的问题，或者访问数据库分区上的表空间或表时发生的问题）将导致装入操作在发生故障的数据库分区上停止运行，但在其余数据库分区上继续运行。装入数据时在数据库分区上发生的错误将导致整个操作失败。

- LOAD_ERRS_ONLY。设置期间在数据库分区上发生的错误将导致整个装入操作失败。如果在装入数据时发生错误，那么装入操作将在出错的数据库分区上停止运行。装入操作将在其余数据库分区上继续运行，直到发生故障或者装入了所有数据为止。在执行装入重新启动操作并成功完成之前，新装入的数据将不可视。

  注: 在同时指定了 LOAD 命令的 ALLOW READ ACCESS 和 COPY YES 选项时，不能使用此方式。
- SETUP_AND_LOAD_ERRS。在此方式下，设置期间或装入数据期间发生的数据库分区级别错误将导致仅在受影响的数据库分区上停止处理装入操作。对于 LOAD_ERRS_ONLY 方式，如果在装入数据时发生分区错误，那么在执行装入重新启动操作并成功完成之前，新装入的数据将不可视。

  注: 在同时指定了 LOAD 命令的 ALLOW READ ACCESS 和 COPY YES 选项时，不能使用此方式。
- NO_ISOLATION。装入操作期间发生的任何错误都会导致装入操作失败。

STATUS_INTERVAL X

X 表示读取多少数据量时发出通知。计量单位是兆字节（MB）。缺省值是 100 MB。有效值是 1 到 4000 的整数。

PORT_RANGE X

X 表示用来创建内部通信套接字的 TCP 端口的范围。缺省范围是 6000 到 6063。如果在调用时定义了 **DB2ATLD_PORTS** 注册表变量的值，那么该值将替换 PORT_RANGE 装入配置选项的值。对于 **DB2ATLD_PORTS** 注册表变量，应该使用以下格式来提供范围:

```
<lower-port-number:higher-port-number>
```

在 CLP 中，格式为:

```
( lower-port-number, higher-port-number )
```

CHECK_TRUNCATION

指定程序应该在输入/输出时检查数据记录截断情况。缺省行为是: 输入/输出时不检查数据截断情况。

MAP_FILE_INPUT X

X 指定分布图的输入文件名。由于此参数指向包含定制分布图的文件，所以，如果使用定制分布图，就必须指定此参数。通过使用 db2gpmap 程序从数据库系统目录表中抽取映射，或者使用 LOAD 命令的 ANALYZE 方式来生成最佳映射，可以创建定制分布图。必须先将使用 ANALYZE 方式生成的映射移至数据库中的每个数据库分区，这样装入操作才能继续运行。

MAP_FILE_OUTPUT X

X 表示分布图的输出文件名。将在发出 LOAD 命令的数据库分区上创建输出文件（假定执行分区操作的数据库分区组包含该数据库分区）。如果在未参与分区的数据库分区（由 PARTITIONING_DBPARTNUMS 定义）上调用 LOAD 命令，那么会在使用 PARTITIONING_DBPARTNUMS 参数定义的第一个数据库分区上创建输出文件。考虑以下分区数据库环境设置:

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

在 serv3 上运行以下 LOAD 命令将在 serv1 上创建分布图。

```
LOAD FROM file OF ASC METHOD L ( ...) INSERT INTO table CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

指定了 ANALYZE 方式时，应该使用此参数。生成最佳分布图（在所有数据库分区之间均匀地分布数据）。如果未指定此参数但指定了 ANALYZE 方式，那么程序将出错并退出。

TRACE X

当您要求复查数据转换过程转储和散列值输出时，指定要跟踪的记录的个数。缺省值为 0。

NEWLINE

当输入数据文件是 ASC 文件（各个记录由换行符定界），并且在 LOAD 命令中指定了 reclen 文件类型修饰符时，使用此选项。当指定了此选项时，将对每个记录检查换行符。还将检查 reclen 文件类型修饰符中指定的记录长度。

DISTFILE X

如果指定了此选项，那么 load 实用程序将生成具有给定名称的数据库分区分布文件。数据库分区分布文件包含 4096 个整数；目标表分布图中的每个条目都有一个对应的整数。此文件中的每个整数都表示所装入输入文件中被分散到相应分布图条目的行数。此信息可以帮助您标识数据偏差，并且还可以帮助您确定是否应该使用实用程序的 ANALYZE 方式来生成表的新分布图。如果未指定此选项，那么 load 实用程序的缺省行为是不生成分布文件。

注: 当指定了此选项时，对装入操作最多使用一个分布代理程序。即使您显式请求多个分布代理程序，也只使用一个。

OMIT_HEADER

指定在分布文件中不应包括分布图头。如果未指定，那么生成头。

RUN_STAT_DBPARTNUM X

如果在 LOAD 命令中指定了 STATISTICS YES 参数，那么将只在一个数据库分区上收集统计信息。此参数指定要收集统计信息的数据库分区。如果值为 -1，或者根本未指定值，那么将在输出数据库分区列表中的第一个数据库分区上收集统计信息。

参考 - 装入

LOAD

将数据装入 DB2 表中。服务器上的数据以文件、磁带或命名管道的形式存在。如果表的 COMPRESS 属性设置为 YES，那么装入的数据将受到每个数据的压缩情况以及在表中已存在字典的数据库分区的支配。

指向 第 199 页的『LOAD 实用程序的文件类型修饰符』的快速链接。

限制

LOAD 实用程序不支持在层次结构级别装入数据。LOAD 实用程序与范围集群表不兼容。

作用域

可通过单个请求向多个数据库分区发出此命令。

权限

为下列其中一项:

- *sysadm*
- *dbadm*
- 对数据库的 LOAD 权限，以及下列特权
  - 以 INSERT 方式、TERMINATE 方式（用于终止先前的装入插入操作）或 RESTART 方式（用于重新启动先前的装入插入操作）调用 LOAD 实用程序时，对表的 INSERT 特权
  - 以 REPLACE 方式、TERMINATE 方式（用于终止先前的装入替换操作）或 RESTART 方式（用于重新启动先前的装入替换操作）调用 LOAD 实用程序时，对表的 INSERT 和 DELETE 特权
  - 作为装入操作使用异常表时，对异常表的 INSERT 特权
- 要将数据装入到包含受保护列的表中，会话授权标识必须拥有允许对该表中所有受保护列执行写访问的 LBAC 凭证。否则，装入将失败并且返回错误（SQLSTATE 5U014）。
- 要将数据装入包含受保护行的表中，会话授权标识必须拥有满足下列条件的安全标号:
  - 它是用于保护表的安全策略的一部分
  - 已将它授予会话授权标识以进行写访问或所有访问

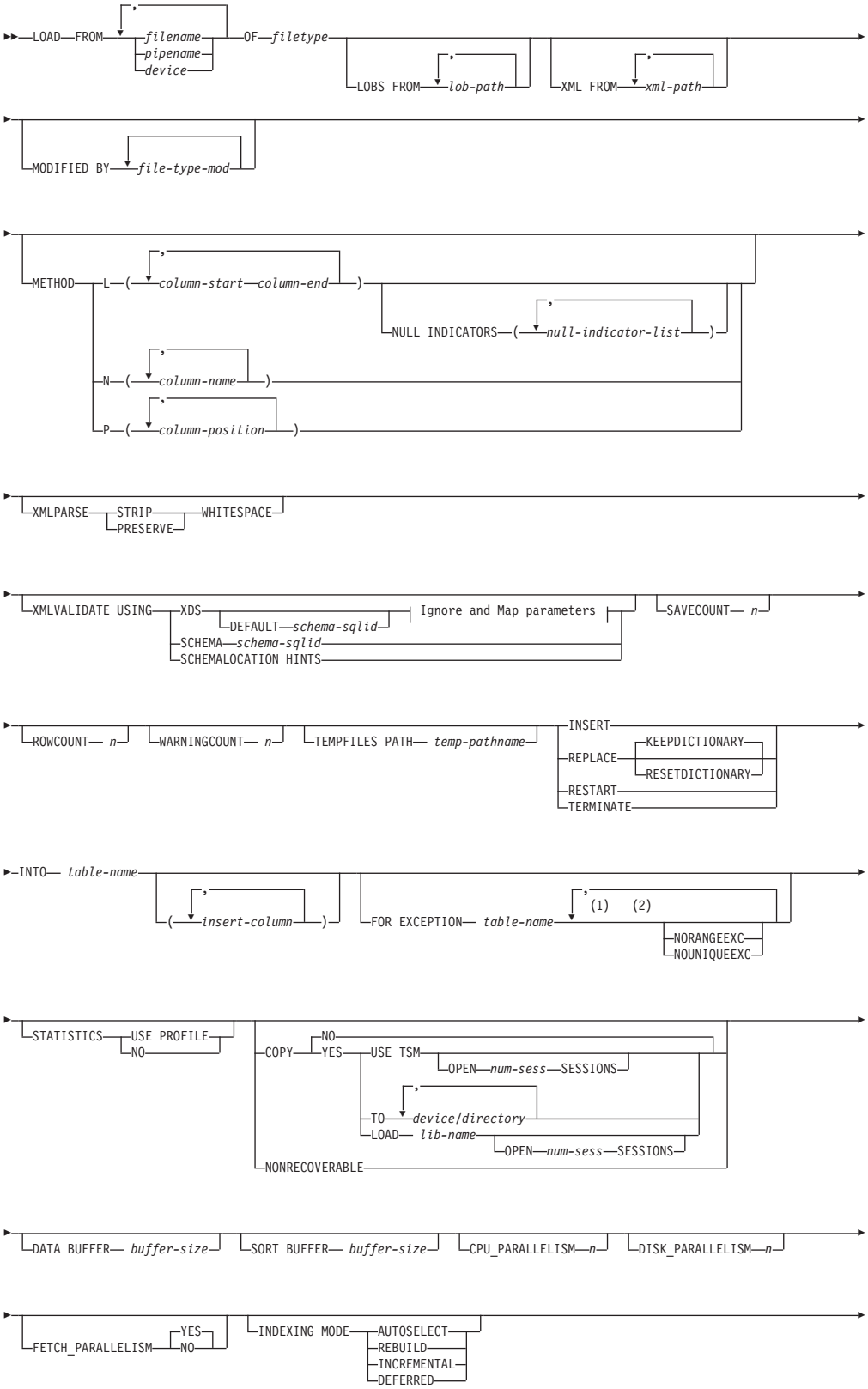
  如果会话授权标识没有这样的安全标号，那么装入将失败并且返回错误（SQLSTATE 5U014）。如果会话授权标识的 LBAC 凭证不允许写入用于保护数据中的已装入行的安全标号，那么此安全标号将用来保护该行。但是，如果用于保护该表的安全策略是使用 CREATE SECURITY POLICY 语句的 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项来创建的，那么不会发生这种情况。在这种情况下，装入将失败并且返回错误（SQLSTATE 42519）。
- 如果指定了 REPLACE 选项，那么会话授权标识必须有权删除该表。

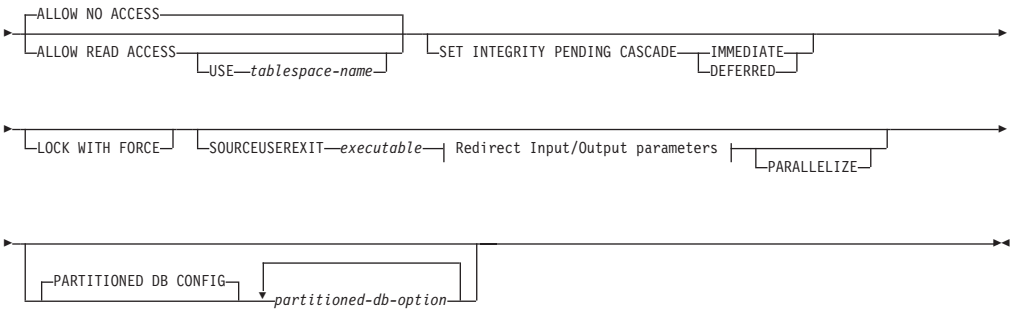
由于所有装入进程（通常还包括所有 DB2 服务器进程）都由实例所有者拥有，并且所有这些进程都使用实例所有者的标识来访问所需的文件，因此，实例所有者必须对输入数据文件具有读访问权。无论谁调用该命令，实例所有者都必须能够读取这些输入数据文件。

必需的连接

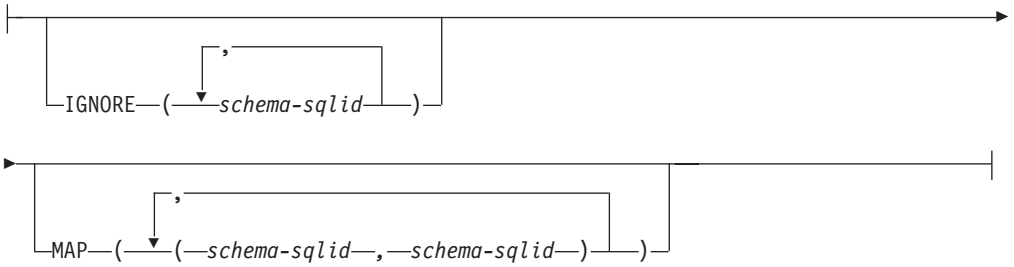
实例。不需要显式连接。如果已经与数据库建立了连接，那么会尝试与本地实例进行隐式连接。

命令语法

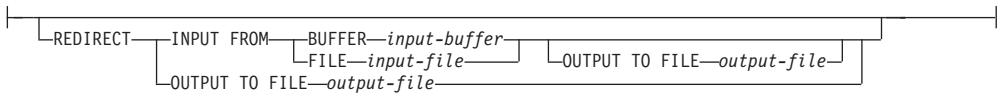

LOAD FROM filename pipename device OF filetype
LOBS FROM lob-path
XML FROM xml-path
MODIFIED BY file-type-mod
METHOD L column-start column-end
NULL INDICATORS null-indicator-list
N column-name
P column-position
XMLPARSE STRIP PRESERVE WHITESPACE
XMLVALIDATE USING XDS DEFAULT schema-sqlid
SCHEMA schema-sqlid
SCHEMALOCATION HINTS
Ignore and Map parameters
SAVECOUNT n
ROWCOUNT n
WARNINGCOUNT n
TEMPFILES PATH temp-pathname
INSERT
REPLACE KEEPDICTIONARY RESETDICTIONARY
RESTART
TERMINATE
INTO table-name
insert-column
FOR EXCEPTION table-name
(1) (2)
NORANGEEXC
NOUNIQUEEXC
STATISTICS USE PROFILE NO
COPY NO YES USE TSM OPEN num-sess SESSIONS
TO device/directory
LOAD lib-name OPEN num-sess SESSIONS
NONRECOVERABLE
DATA BUFFER buffer-size
SORT BUFFER buffer-size
CPU_PARALLELISM n
DISK_PARALLELISM n
FETCH_PARALLELISM YES NO
INDEXING MODE AUTOSELECT REBUILD INCREMENTAL DEFERRED



Ignore and Map parameters:



Redirect Input/Output parameters:



注:

1 这些关键字可以按任意顺序出现。

2 这些关键字中的每一个都只能出现一次。

命令参数

FROM *filename* | *pipename* | *device*

注:

1. 如果通过*使用 ADMIN_CMD 过程的 EXPORT 命令*将数据导出到文件，那么数据文件归受防护用户标识所有。实例所有者通常不能访问此文件。要从 CLP 或 ADMIN_CMD 过程运行 LOAD 命令，实例所有者标识必须能够访问数据文件，以保证将对数据文件的读访问权授予实例所有者。
2. 如果多个 IXF 文件在物理位置上是分隔开的，但在逻辑上是一个文件，那么支持装入这些文件中的数据。如果这些文件在逻辑上和物理位置上都是分开的，那么不支持装入这些文件中的数据。（如果多个物理文件都是通过调用一次 EXPORT 命令来创建的，那么在逻辑上会将这些文件视为是一个文件。）

OF *filetype*

指定数据的格式:

- ASC（非定界 ASCII 格式）。

- DEL（定界 ASCII 格式）。
- IXF（PC 版本的集成交换格式）是一种专门供 DB2 使用的二进制格式。
- CURSOR（对 SELECT 或 VALUES 语句声明的游标）。

LOBS FROM *lob-path*

包含要装入的 LOB 值的数据文件的路径。该路径必须以斜杠（/）结尾。LOB 数据文件的名称存储在将装入到 LOB 列的那列的主数据文件（ASC、DEL 或 IXF）中。最多可指定 999 个路径。这将隐式激活 LOBSINFILE 行为。

当将此选项与 CURSOR 文件类型一起指定时，将忽略此选项。

MODIFIED BY *file-type-mod*

指定文件类型修饰符选项。请参阅 第 199 页的『LOAD 实用程序的文件类型修饰符』。

METHOD

L

指定要将数据装入到的列的起始列号和结束列号。列号就是与一个数据行的开头的字节偏移量。它是从 1 开始计数的。此方法只能用于 ASC 文件，并且对于该文件类型是唯一有效的方法。

NULL INDICATORS *null-indicator-list*

仅当指定了 METHOD L 参数时（即，输入文件是 ASC 文件），才能使用此选项。空指示符列表是用于指定每个空指示符字段的正整数的列表，各个正整数之间用逗号隔开。列号是空指示符字段与一个数据行的开头之间的字节偏移量。对于 METHOD L 参数中定义的每个数据字段，在空指示符列表中必须有一个其对应的条目。如果列号为 0，那么表示相应的数据字段中始终包含数据。

如果空指示符列中的值为 Y，那么表示列数据为 NULL。如果空指示符列中的值是除了 Y 之外的任何字符，那么表示列数据不是 NULL，并且将装入由 METHOD L 选项指定的列数据。

可以使用 MODIFIED BY 选项来更改空指示符。

N

指定要装入的数据文件中的列名。这些列名的大小写必须与系统目录中相应名称的大小写相匹配。不可空的每个表列在 METHOD N 列表中都应该具有相应的条目。例如，给定数据字段 F1、F2、F3、F4、F5 和 F6，以及表列 C1 INT、C2 INT NOT NULL、C3 INT NOT NULL 和 C4 INT，那么 `method N (F2, F1, F4, F3)` 是有效请求，而 `method N (F2, F1)` 是无效请求。此方法只能用于文件类型 IXF 或 CURSOR。

P

指定要装入的输入数据字段的字段编号（从 1 开始计数）。不可空的每个表列在 METHOD P 列表中都应该具有相应的条目。例如，给定数据字段 F1、F2、F3、F4、F5 和 F6，以及表列 C1 INT、C2 INT NOT NULL、C3 INT NOT NULL 和 C4 INT，那么 `method P (2, 1, 4, 3)` 是有效请求，而 `method P (2, 1)` 是无效请求。此方法只能用于文件类型 IXF、DEL 或 CURSOR，并且对于 DEL 文件类型是唯一有效的方法。

XML FROM *xml-path*

指定一个或多个包含 XML 文件的路径。XDS 包含在主数据文件（ASC、DEL 或 IXF）中将装入到 XML 列中的那一列中。

XMLPARSE

指定如何解析 XML 文档。如果未指定此选项，那么将由 CURRENT XMLPARSE OPTION 专用寄存器的值来确定 XML 文档的解析行为。

STRIP WHITESPACE

指定在解析 XML 文档时要去掉空格。

PRESERVE WHITESPACE

指定在解析 XML 文档时不去掉空格。

XMLVALIDATE

指定在适当的情况下将针对某一模式来验证 XML 文档。

USING XDS

将针对由主数据文件中的 XML 数据说明符（XDS）标识的 XML 模式来验证 XML 文档。缺省情况下，如果使用 USING XDS 子句调用了 XMLVALIDATE 选项，那么将由 XDS 的 SCH 属性来确定用来执行验证的模式。如果 XDS 中不存在 SCH 属性，那么除非缺省模式是由 DEFAULT 子句指定的，否则将不会进行模式验证。

可以使用 DEFAULT、IGNORE 和 MAP 子句来修改模式确定行为。这三个可选子句直接应用于 XDS 的指定，但是它们不会互相应用。例如，如果由于 DEFAULT 子句指定了某一模式而选择了该模式，那么，即使 IGNORE 子句也指定了该模式，该模式仍不会被忽略。类似，如果由于将某一模式指定为 MAP 子句对的第一部分而选择了该模式，那么，即使在另一个 MAP 子句对的第二部分中也指定了该模式，该模式仍不会被重新映射。

USING SCHEMA *schema-sqlid*

将针对具有指定的 SQL 标识的 XML 模式来验证 XML 文档。在这种情况下，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

USING SCHEMALOCATION HINTS

将针对由源 XML 文档中的 XML 模式位置提示标识的模式来验证 XML 文档。如果在 XML 文档中找不到 schemaLocation 属性，那么将不执行验证。指定 USING SCHEMALOCATION HINTS 子句时，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

请参阅下面的 XMLVALIDATE 选项示例。

IGNORE *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。如果 SCH 属性标识了一种或多种模式，那么 IGNORE 子句指定这些模式中要忽略的模式列表。如果 SCH 属性存在于已装入的 XML 文档的 XML 数据说明符中，并且由 SCH 属性标识的模式包含在要忽略（IGNORE）的模式列表中，那么将不会对已装入的 XML 文档进行模式验证。

注:

如果某一模式是在 IGNORE 子句中指定的，那么该模式不能存在于 MAP 子句中模式对的左边。

IGNORE 子句仅适用于 XDS。如果 IGNORE 子句指定了由 MAP 子句映射的模式，那么以后不会忽略该模式。

DEFAULT *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。通过 DEFAULT 子句指定的模式标识：当已装入的 XML 文档的 XML 数据说明符（XDS）不包含用于标识 XML 模式的 SCH 属性时要用于验证的模式。

DEFAULT 子句优先于 IGNORE 和 MAP 子句。如果 XDS 满足 DEFAULT 子句，那么将忽略 IGNORE 和 MAP 规范。

MAP *schema-sqlid*

仅当指定了 USING XDS 参数时，才能使用此选项。使用 MAP 子句来指定要使用的替代模式，这些模式将取代由已装入的每个 XML 文档的 XML 数据说明符（XDS）的 SCH 属性所指定的那些模式。MAP 子句指定一个或多个模式对的列表，而每个模式对都表示一个模式与另一个模式之间的映射。模式对中的第一个模式表示 XDS 中的 SCH 属性引用的模式。模式对中的第二个模式表示应该用来执行模式验证的模式。

如果某一模式存在于 MAP 子句中的模式对的左边，那么不能在 IGNORE 子句中也指定该模式。

一旦应用了模式对映射，其结果就是最终结果。映射操作不是过渡操作，因此，以后不会将所选择的模式应用于另一个模式对映射。

不能多次映射同一模式，这就意味着该模式不会多次出现在一个模式对的左边。

SAVECOUNT *n*

指定 LOAD 实用程序在每隔 *n* 行之后就要建立一致点。此值被转换为页面计数，并且向上取整为扩展数据块大小的间隔。由于在每个一致点都会发出消息，因此，如果将使用 LOAD QUERY 来监视装入操作，那么应选择此选项。如果 *n* 的值不是足够大，那么在每个一致点执行的活动的同步将影响性能。

缺省值为 0，这意味着除非确实需要建立一致点，否则将不会建立任何一致点。

当将此选项与 CURSOR 文件类型一起指定时，将忽略此选项。

ROWCOUNT *n*

指定要装入的文件中的 *n* 个物理记录。允许用户只装入文件中前面的 *n* 行。

WARNINGCOUNT *n*

在发出 *n* 个警告后停止装入操作。如果希望不产生警告，那么设置此参数，但是需要验证使用的是正确的文件和表。如果未正确指定装入文件或目标表，那么 LOAD 实用程序将对它试图装入的每一行生成一个警告，这将导致装入失败。如果 *n* 为 0，或者未指定此选项，那么无论发出多少个警告，都将继续执行装入操作。如果因达到了警告数的阈值而停止了装入操作，那么可以采用 RESTART 方式开始执行另一个装入操作。装入操作将自动从最后一个一致点继续执行。或者，可以采用 REPLACE 方式启动另一个装入操作，并从输入文件开头开始装入。

TEMPFILES PATH *temp-pathname*

指定在执行装入操作期间创建临时文件时要使用的路径（相对于服务器数据库分区而言，此路径应该是标准路径）的名称。

临时文件将占用文件系统空间。有时，此空间需求是相当大的。以下是估计应该为所有临时文件分配的文件系统空间量:

- LOAD 实用程序生成的每条消息需要 136 字节。
- 如果数据文件包含长整型字段数据或 LOB，那么需要 15 KB 的开销。如果指定了 INSERT 选项，并且表中已经具有大量的长整型字段或 LOB 数据，那么需要的空间量会显著增加。

INSERT

执行 LOAD 实用程序时可以采用的四种方式之一。将已装入的数据添加至表，但不更改现有表数据。

REPLACE

执行 LOAD 实用程序时可以采用的四种方式之一。从表中删除现有的所有数据，然后插入已装入的数据。表定义和索引定义不会发生更改。如果在层次结构之间移动数据时使用了此选项，那么只能替换整个层次结构的数据，而不能替换单个子表的数据。

KEEPDICTIONARY

在执行 LOAD REPLACE 操作期间将保留现有压缩字典。如果表 COMPRESS 属性是 YES，那么将使用执行装入调用前存在的字典来压缩新替换的数据。如果在此之前，表中不存在任何字典，那么只要表 COMPRESS 属性是 YES，就会使用要替换到表中的数据来构建新字典。在这种情况下，构建压缩字典所需的数据量受 ADC 策略的约束。此数据将以未压缩的形式填充到表中。将该字典插入表中后，余下要装入的数据将使用此字典进行压缩。这是缺省参数。请参阅下面的表 1 以获取相关摘要。

表 27. LOAD REPLACE KEEPDICTIONARY

压缩	存在字典	结果
Y	Y	保留字典；所有输入行都将使用现有字典进行压缩。
Y	N	只有存在足够的用户数据时，才将新字典插入表中；在构建字典后，余下的行将使用该字典进行压缩。
N	Y	保留字典；不压缩所有输入行。
N	N	无影响；不压缩所有行。

RESETDICTIONARY

如果表 COMPRESS 属性是 YES，那么此伪指令指示 LOAD REPLACE 处理操作为表数据对象构建新字典。如果 COMPRESS 属性是 NO，而表中已存在字典，那么会将该字典除去并且不会将任何新字典插入该表中。可以构建仅包含一条用户记录的压缩字典。如果装入的数据集大小为 0 且预先存在字典，那么将不保留该字典。使用此伪指令构建字典所需的数据量不受 ADC 策略的约束。请参阅下面的表 2 以获取相关摘要。

表 28. LOAD REPLACE RESETDICTIONARY

压缩	存在字典	结果
Y	Y	构建新字典*；构建字典后，将使用该字典压缩余下要装入的行。
Y	N	构建新字典；构建字典后，将使用该字典压缩余下的行。
N	Y	除去字典；不压缩所有输入行。

表 28. LOAD REPLACE RESETDICTIONARY (续)

压缩	存在字典	结果
N	N	无影响；不压缩所有行。

* 如果存在字典且启用了压缩属性，但没有任何记录可供装入到表分区中，那么无法构建新字典且 RESETDICTIONARY 操作将不保留现有字典。

TERMINATE

执行 LOAD 实用程序时可以采用的四种方式之一。终止先前已中断的装入操作，并将该操作回滚到开始执行它时的时间点，即使已经超过了一致点亦如此。该操作中涉及到的任何表空间的状态将恢复为正常状态，并且会使所有表对象保持一致（索引对象可能会被标记为无效。在这种情况下，下一次进行访问时将自动重建索引）。如果终止的装入操作是 LOAD REPLACE，那么在执行 LOAD TERMINATE 操作后，表将被截断为一个空表。如果终止的装入操作是 LOAD INSERT，那么在执行 LOAD TERMINATE 操作后，表将保留它的所有原始记录。请参阅下面的表 3 以获取字典管理的摘要。

LOAD TERMINATE 选项不会使表空间脱离“备份暂挂”状态。

RESTART

执行 LOAD 实用程序时可以采用的四种方式之一。重新启动先前已中断的装入操作。装入操作将自动从装入、构建或删除阶段中的最后一个一致点继续执行。请参阅下面的表 4 以获取字典管理的摘要。

INTO *table-name*

指定要将数据装入到的数据库表。此表不能是系统表或者已声明临时表。可指定别名，或者指定标准表名或非限定表名。标准表名的格式为 schema.tablename。如果指定了非限定表名，那么将使用 CURRENT SCHEMA 语句来限定表。

insert-column

指定要将数据插入到的表列。

LOAD 实用程序将无法解析名称中包含一个或多个空格的那些列。例如，

将因 Int 4 列而失败。解决方案是将这些列名用双引号引起来:

FOR EXCEPTION *table-name*

指定要将错误行复制到的异常表。将复制违反唯一索引或主键索引的任何行。如果指定了非限定表名，那么将使用 CURRENT SCHEMA 语句来限定表。

不会将写入异常表的信息写入转储文件。在分区数据库环境中，必须为定义了装入表的那些数据库分区定义异常表。除此之外，转储文件将包含无法装入的行，这是因为这些行无效或者具有语法错误。

NORANGEEXC

指示如果某行因违反范围而被拒绝，那么不会将该行插入到异常表中。

NOUNIQUEEXC

指示如果某行因违反唯一约束而被拒绝，那么不会将该行插入到异常表中。

STATISTICS USE PROFILE

指示 LOAD 实用程序在装入操作期间根据为此表定义的概要文件来收集统计信息。必须在执行装入前创建此概要文件。此概要文件是使用 RUNSTATS 命令

创建的。如果此概要文件不存在，但是您指示装入操作根据此概要文件来收集统计信息，那么会返回警告，并且不会收集统计信息。

STATISTICS NO

指定将不会收集统计信息，目录中的统计信息也不会改变。这是缺省值。

COPY NO

指定如果启用了正向恢复（即，打开了 *logretain* 或 *userexit*），那么表所在的表空间将被置于“备份暂挂”状态。COPY NO 选项还会将表空间状态变为“正在装入”表空间状态。这是一种瞬态状态，当装入操作完成或者异常中止时，此状态就会消失。在备份表空间或者备份整个数据库后，才能更新或删除表空间中的任何表所包含的数据。但是，可以使用 SELECT 语句来访问任何表中的数据。

对可恢复的数据库执行指定了 COPY NO 选项的 LOAD 命令时，会将表空间置于“备份暂挂”状态。例如，执行指定了 COPY NO 和 INDEXING MODE DEFERRED 选项的 LOAD 命令时，将需要刷新索引。对表的某些查询可能需要进行索引扫描，并且在刷新索引前将不会成功。如果索引位于一个处于“备份暂挂”状态的表空间中，那么无法刷新该索引。在这种情况下，在执行备份前将不允许访问表。当查询访问索引时，数据库将自动完成索引刷新。如果未指定 COPY NO、COPY YES 或 NONRECOVERABLE 之一，并且数据库是可恢复的（启用了 **logretain** 或 **logarchmeth1**），那么缺省值为 COPY NO。

COPY YES

指定将保存已装入数据的副本。如果禁用了正向恢复（即，关闭了 *logretain* 和 *userexit*），那么此选项无效。

USE TSM

指定将使用 Tivoli Storage Manager（TSM）来存储副本。

OPEN *num-sess* **SESSIONS**

将与 TSM 或供应商产品一起使用的 I/O 会话数。缺省值为 1。

TO *device/directory*

指定将创建复制映像的设备或目录。

LOAD *lib-name*

共享库（在 Windows 操作系统上为 DLL）的名称，该共享库包含要使用的供应商备份与复原 I/O 函数。也可以包含完整路径。如果未提供完整路径，将缺省为用户出口程序所在的路径。

NONRECOVERABLE

指定装入事务将标记为不可恢复，并且后续前滚操作不能将其恢复。Rollforward 实用程序将跳过该事务，并且会将装入数据的表标记为“无效”。该实用程序还将忽略该对表执行的任何后续事务。在完成前滚操作后，只能删除这样的表，或者从完成不可恢复的装入操作后的落实点后生成的备份（完整或表空间）来复原该表。

如果指定了此选项，在装入操作完成后就不会将表空间置于“备份暂挂”状态，并且在装入操作执行期间不必创建所装入数据的副本。如果未指定 COPY NO、COPY YES 或 NONRECOVERABLE 之一，并且数据不可恢复（未启用 **logretain** 或 **logarchmeth1**），那么缺省值为 NONRECOVERABLE。

WITHOUT PROMPTING

指定数据文件列表中包含要装入的所有文件，并且所列示的设备或目录足以满

足整个装入操作的需求。如果找不到下一个输入文件，或者在完成装入操作前填充了复制目标，那么装入操作将失败，并且表将保持处于装入暂挂状态。

DATA BUFFER *buffer-size*

指定要用作用于传送实用程序中数据的缓存空间的 4 KB 页数（不考虑并行度）。如果指定值小于算术最小值，那么将使用需要的最少资源，但不会返回警告。

此内存是直接从实用程序堆中分配的，可通过数据库配置参数 *util_heap_sz* 来修改此内存大小。

如果未指定值，那么实用程序在运行时将计算智能缺省值。该缺省值取决于在实例化装入程序时实用程序堆中的可用空间以及表的某些特征。

SORT BUFFER *buffer-size*

此选项指定一个在装入操作期间将覆盖 SORTHEAP 数据库配置参数的值。仅当装入具有索引的表以及 INDEXING MODE 参数未指定为 DEFERRED 时，此选项才有效。为此选项指定的值不能超过 SORTHEAP 的值。此参数对于在不更改 SORTHEAP 值的情况下调整在装入具有许多索引的表时所使用的排序内存是很有用的，这还会影响常规查询处理。

CPU_PARALLELISM *n*

指定当构建表对象时，LOAD 实用程序为了解析、转换和格式化记录而创建的进程数或线程数。此参数旨在利用分区内并行性。当装入已预先排序的数据时，此参数特别有用，这是因为将保留源数据中的记录顺序。如果此参数的值为零或者尚未指定，那么 LOAD 实用程序在运行时将使用智能缺省值（它通常基于可用 CPU 的数目）。

注:

1. 如果对包含 LOB 或 LONG VARCHAR 字段的表使用此参数，那么无论系统 CPU 的数目或者用户指定的值是多少，此参数的值都将为 1。
2. 如果对 SAVECOUNT 参数指定一个较小的值，那么会导致装入程序执行许多 I/O 操作来同时清空数据和表元数据。当 CPU_PARALLELISM 大于 1 时，清仓操作是异步执行的，并且允许装入程序利用 CPU。当 CPU_PARALLELISM 设置为 1 时，装入程序在一致点期间将等待 I/O 操作。对于装入操作，尽管只有一个 CPU，但是如果将 CPU_PARALLELISM 设置为 2 并将 SAVECOUNT 设置为 10000，会比将 CPU_PARALLELISM 设置为 1 时完成得更快。

DISK_PARALLELISM *n*

指定 LOAD 实用程序为了将数据写入表空间容器而创建的进程数或线程数。如果未指定值，那么该实用程序将根据表空间容器数和表的特征来选择智能缺省值。

FETCH_PARALLELISM YES | NO

当执行从游标装入（该游标是使用 DATABASE 关键字声明的），或者当使用 API `sqlu_remotefetch_entry` 介质条目，并且此选项设置为 YES 时，LOAD 实用程序将尝试从远程数据源并行访存（如果可能的话）。如果设置为 NO，那么不会执行并行访存。缺省值为 YES。有关更多信息，请参阅*使用 CURSOR 文件类型来移动数据*。

INDEXING MODE

指定 LOAD 实用程序是要重建索引还是以增量方式扩展索引。有效值为:

AUTOSELECT

LOAD 实用程序将自动决定是使用 REBUILD 还是使用 INCREMENTAL 方式。应根据装入的数据量和索引树的深度来作出决定。与索引树深度相关的信息存储在索引对象中。不需要执行 RUNSTATS 来填充此信息。AUTOSELECT 是建立索引的缺省方式。

REBUILD

将重建所有索引。实用程序必须具有足够的资源来对旧的表数据和追加的表数据的所有索引键部分进行排序。

INCREMENTAL

将为索引扩充新数据。此方法将消耗索引可用空间。它只需要足够的排序空间来为已插入的记录追加索引键。只有在下列情况下才支持此方法: 当装入操作开始时，索引对象有效并且可访问（例如，在执行指定了 DEFERRED 方式的装入操作后，它将立即变得无效）。如果指定了此方式，但是因索引状态而不受支持，那么会返回警告，并且装入操作将继续采用 REBUILD 方式。同样，如果在装入构建阶段开始执行“重新启动装入”操作，那么不支持 INCREMENTAL 方式。

当满足下列所有条件时，就不支持以 INCREMENTAL 方式建立索引:

- 指定了 LOAD COPY 选项（指定了 USEREXIT 或 LOGRETAIN 选项的 *logarchmeth1*）。
- 表位于 DMS 表空间中。
- 索引对象位于一个与属于正在装入的表的其他表对象共享的表空间中。

要绕过此限制，建议您将索引放在一个单独的表空间中。

DEFERRED

如果指定此方式，那么 LOAD 实用程序将不会尝试创建索引。索引将被标记为需要刷新。首次访问与装入操作不相关的这样的索引可能会强制执行重建，或者在重新启动数据库时可能会重建索引。这种方法要求为最大索引的所有键部分提供足够的排序空间。后续用于构造索引的总时间比在 REBUILD 方式下所需要的时间更长。因此，在以 DEFERRED 方式建立索引的情况下执行多个装入操作时，（从性能方面考虑）建议您在最后一个装入操作中执行索引重建，而不允许在第一次进行非装入访问时重建索引。

仅支持对具有非唯一索引的表以 DEFERRED 方式建立索引，以便在完成装入操作后不会持久保留在装入期间插入的重复键。

ALLOW NO ACCESS

LOAD 实用程序将锁定目标表，以便在装入期间进行互斥访问。在装入期间，表的状态将设置为“正在装入”。ALLOW NO ACCESS 是缺省行为。对于 LOAD REPLACE，它是唯一有效的选项。

当表存在约束时，表的状态将设置为“设置完整性暂挂”以及“正在装入”。必须使用 SET INTEGRITY 语句来使表脱离“设置完整性暂挂”状态。

ALLOW READ ACCESS

LOAD 实用程序将锁定采用共享方式的目标表。表的状态将设置为“正在装入”和“读访问”。在装入表时，阅读器可以访问数据的非增量部分。换句话说，表的阅读器将能够访问在开始装入前就已存在的数据，而在完成装入前，正在

装入的数据将不可用。ALLOW READ ACCESS 装入的 LOAD TERMINATE 或 LOAD RESTART 可以使用此选项；而 ALLOW NO ACCESS 装入的 LOAD TERMINATE 或 LOAD RESTART 不能使用此选项。而且，如果需要重建目标表的索引，那么此选项无效。

当表存在约束时，表的状态将设置为“设置完整性暂挂”以及“正在装入”和“读访问”。装入结束时，“正在装入”这种表状态将不再存在，但是“设置完整性暂挂”和“读访问”这两种表状态将继续存在。必须使用 SET INTEGRITY 语句来使表脱离“设置完整性暂挂”状态。当表处于“设置完整性暂挂”和“读访问”状态时，阅读器仍然可以访问数据的非增量部分，但是在完成 SET INTEGRITY 语句前，将仍然不可访问数据的新增（增量）部分。用户可以对同一个表执行多次装入而无须发出 SET INTEGRITY 语句。但是，在发出 SET INTEGRITY 语句前，只有原始（已检查的）数据仍然可视。

ALLOW READ ACCESS 还支持下列修饰符:

USE *tablespace-name*

如果正在重建索引，那么在 *tablespace-name* 表空间中构建了索引的影子副本，并且在 INDEX COPY PHASE 期间在装入结束时复制到了原始表空间中。只能对系统临时表空间使用此选项。如果未指定此选项，那么将在索引对象所在的同一表空间中创建影子索引。如果在索引对象所在的同一表空间中创建了影子副本，那么会立即复制影子索引对象来覆盖旧的索引对象。如果影子副本与索引对象不在同一个表空间中，那么将执行物理复制。这可能要执行大量的 I/O 操作和花费大量时间。在 INDEX COPY PHASE 期间，当装入结束时，如果表处于脱机状态，那么会进行复制。

在未使用此选项的情况下，将在原始索引所在的表空间中构建影子索引。由于原始索引和影子索引在缺省情况下同时位于同一个表空间中，因此，可能没有足够的空间用来将这两种索引保存在同一个表空间中。但是，如果使用此选项，就可以确保为索引保留足够的表空间。

如果用户不指定 INDEXING MODE REBUILD 或 INDEXING MODE AUTOSELECT，那么将忽略此选项。如果选择了 INDEXING MODE AUTOSELECT，并且 LOAD 实用程序选择以增量方式更新索引，那么也将忽略此选项。

SET INTEGRITY PENDING CASCADE

如果 LOAD 实用程序将表置于“设置完整性暂挂”状态，那么 SET INTEGRITY PENDING CASCADE 选项允许用户指定是否将已装入的表的“设置完整性暂挂”状态立即级联至所有后代（包括派生外键表、派生立即具体化查询表和派生立即登台表）。

IMMEDIATE

指示“设置完整性暂挂”状态将立即扩展至所有派生外键表、派生立即具体化查询表和派生立即登台表。对于 LOAD INSERT 操作，即使指定了 IMMEDIATE 选项，也不会将“设置完整性暂挂”状态扩展至派生外键表。

稍后，当（使用 SET INTEGRITY 语句的 IMMEDIATE CHECKED 选项）检查已装入的表是否存在约束违例时，先前处于“设置完整性暂挂读访问”状态的派生外键表将被置于“设置完整性暂挂无访问”状态。

DEFERRED

指示只有已装入的表才将处于“设置完整性暂挂”状态。而派生外键表、派生立即具体化查询表和派生立即登台表的状态将保持不变。

稍后，当（使用 SET INTEGRITY 语句的 IMMEDIATE CHECKED 选项）来检查派生外键表的父表是否发生了约束违例时，这些派生外键表可能会隐式地处于“设置完整性暂挂”状态。当检查派生立即具体化查询表和派生立即登台表的其中一个基础表是否发生了完整性违例时，这些父表可能会隐式地处于“设置完整性暂挂”状态。将发出警告（SQLSTATE 01586），指出从属表已经处于“设置完整性暂挂”状态。请参阅 SQL Reference 中 SET INTEGRITY 语句的“注释”部分，以了解这些派生表何时将处于“设置完整性暂挂”状态。

如果未指定 SET INTEGRITY PENDING CASCADE 选项:

- 则只有已装入的表才将处于“设置完整性暂挂”状态。派生外键表、派生立即具体化查询表和派生立即登台表的状态将保持不变，稍后，当检查已装入的表是否存在约束违例时，上述表可能会隐式地处于“设置完整性暂挂”状态。

如果 LOAD 实用程序并未将目标表置于“设置完整性暂挂”状态，那么说明忽略了 SET INTEGRITY PENDING CASCADE 选项。

LOCK WITH FORCE

在装入过程中，实用程序将发生各种锁定（包括表锁定）。当产生锁定时，将不采用等待策略，等待有可能会超时；此选项将允许 LOAD 实用程序强制关闭对目标表拥有相冲突锁定的其他应用程序。LOAD 实用程序不会强制关闭对系统目录表拥有相冲突锁定的应用程序。将回滚已强制关闭的应用程序，并释放 LOAD 实用程序需要的锁定。LOAD 实用程序随后就可以继续执行了。此选项与 FORCE APPLICATIONS 命令需要相同的权限（SYSADM 或 SYSCTRL）。

在装入操作开始时，ALLOW NO ACCESS 装入可能会强制挂起相冲突的锁定的应用程序。在开始装入时，LOAD 实用程序可能会强制要尝试查询或修改表的应用程序。

在装入操作开始或结束时，ALLOW READ ACCESS 装入可能会强制挂起相冲突的锁定的应用程序。在开始装入时，LOAD 实用程序可能会强制要尝试修改表的应用程序。在装入操作结束时，LOAD 实用程序可能会强制要尝试查询或修改表的应用程序。

SOURCEUSEREXIT*executable*

指定一个可执行文件的文件名，将调用该文件来为实用程序输入数据。

REDIRECT

INPUT FROM

BUFFER *input-buffer*

在 *input-buffer* 中指定的字节流被传递给用于执行给定可执行文件的进程的 STDIN 文件描述符中。

FILE *input-file*

此客户端文件的内容被传递给用于执行给定可执行文件的进程的 STDIN 文件描述符中。

OUTPUT TO

FILE *output-file*

STDOUT 和 STDERR 文件描述符被捕获至指定的标准服务器端文件。

PARALLELIZE

通过同时调用多个用户出口进程来提高进入 LOAD 实用程序的数据吞吐量。此选项仅适用于多分区数据库环境，在单一分区数据库环境中将被忽略。

有关更多信息，请参阅*使用定制应用程序（用户出口）移动数据*。

PARTITIONED DB CONFIG *partitioned-db-option*

允许您装入到一个分布在多个数据库分区中的表。PARTITIONED DB CONFIG 参数允许您指定特定于分区数据库的配置选项。*partitioned-db-option* 值可以是下列任何一项:

```
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

在*为分区数据库环境装入配置选项*中提供了对这些选项的详细描述。

RESTARTCOUNT

保留参数。

USING *directory*

保留参数。

从 XML 文档装入数据的示例

从 XML 文档装入数据

示例 1

用户构造了数据文件，该文件包含了用于描述那些要插入表中的文档的 XDS 字段。它看起来可能类似如下:

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

对于第一行，XML 文档由文件 `file1.xml` 标识。注意，由于字符定界符是双引号字符，而 XDS 中存在双引号，因此 XDS 中包含的双引号是两个。对于第二行，XML 文档由文件 `file2.xml` 标识，从字节偏移量为 23 的位置开始，长度为 45 字节。

示例 2

用户对 XML 列发出了不带任何解析或验证选项的装入命令，并且成功地装入了数据:

```
LOAD FROM data.del of DEL INSERT INTO mytable
```

从游标装入 XML 数据

从游标装入数据与使用规则关系列类型一样。用户具有两个表（T1 和 T2），每个表都包含单个称为 C1 的 XML 列。要从 T1 装入到 T2，用户应先声明游标:

```
DECLARE X1 CURSOR FOR SELECT C1 FROM T1;
```

接着，用户可发出使用游标类型的 LOAD:

```
LOAD FROM X1 of CURSOR INSERT INTO T2
```

将特定于 XML 的 LOAD 选项装入到该游标类型与从文件装入一样。

使用说明

- 数据的装入顺序与在输入文件中的出现顺序相同。如果需要使用特定顺序，那么在尝试装入前应该对数据进行排序。如果不需要保留数据源顺序，请考虑使用 ANYORDER 文件类型修饰符（在下面的 *LOAD 实用程序的文件类型修饰符*部分作了描述）。
- LOAD 实用程序将根据现有定义来构建索引。异常表用来处理重复的唯一键。该实用程序不会强制引用完整性、执行约束检查或者更新依赖于正在装入的表的具体化查询表。包含参考约束或检查约束的表将处于“设置完整性暂挂”状态。使用 REFRESH IMMEDIATE 定义的、并且依赖于正在装入的表的总结表也将处于“设置完整性暂挂”状态。发出 SET INTEGRITY 语句以使这些表脱离“设置完整性暂挂”状态。不能对复制的具体化查询表执行装入操作。
- 如果表上存在集群索引，那么应该在执行装入前按集群索引对数据进行排序。但是，在将数据装入到多维集群（MDC）表前，不需要对数据进行排序。
- 如果在装入到受保护的表时指定了异常表，那么会将受到无效安全标号保护的任何行发送至该表。这可能会允许对异常表具有访问权的用户访问他们通常无权访问的数据。为了提高安全性，您在将对异常表的访问权授予用户时应慎重；在修复了每一行并将它复制到要装入的表中后就立即删除该行；并且在使用异常表完毕后就立即将它删除。
- 采用内部格式的安全标号可能包含换行符。如果装入使用 DEL 文件格式的文件，那么这些换行符可能会被误认为定界符。如果发生此问题，请通过在 LOAD 命令中指定文件类型修饰符 delprioritychar 来对定界符使用较旧的缺省优先级。
- 要执行使用 CURSOR 文件类型的装入操作（并且 DATABASE 关键字是在执行 DECLARE CURSOR 命令期间指定的），用来对当前连接至的数据库（用于装入）进行认证的用户标识和密码，将用于对源数据库（由 DECLARE CURSOR 命令的 DATABASE 选项指定）进行认证。如果没有指定用于连接至正在装入的数据库的用户标识或密码，那么在执行 DECLARE CURSOR 命令期间必须指定源数据库的用户标识和密码。
- 支持装入多部件 PC/IXF 文件，该文件的各个部件是从 Windows 系统复制到 AIX 系统的。所有文件的名称都必须在 LOAD 命令中指定。例如，`LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1`。不支持从逻辑上分割的 PC/IXF 文件装入到 Windows 操作系统。
- 重新启动失败的 LOAD 时，该行为将遵循现有行为，在现有行为中，会强制 BUILD 阶段将 REBUILD 方式用于索引。

LOAD TERMINATE 和 LOAD RESTART 字典管理的摘要

下图总结了 TERMINATE 伪指令下 LOAD 处理的压缩字典管理行为。

表 29. LOAD TERMINATE 字典管理

表 COMPRESS 属性	装入前存在字典吗?	TERMINATE: LOAD REPLACE KEEPDICTIONARY 或 LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	保留现有字典。	不保留。
YES	NO	不保留。	不保留。
NO	YES	保留现有字典。	不保留。
NO	NO	不执行任何操作。	不执行任何操作。

LOAD RESTART 将截断表直至达到最后一个一致点。如果获取最后一个 LOAD 一致点时表中存在压缩字典，那么在执行 LOAD RESTART 处理过程中，表中将存在压缩字典。在那种情况下，LOAD RESTART 将不会创建新字典。请参阅下面的表 4 以获取可能情况的摘要。

表 30. LOAD RESTART 字典管理

表 COMPRESS 属性	装入一致点前是否存在字典?	RESTART: LOAD REPLACE KEEPDICTIONARY 或 LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	保留现有字典。	保留现有字典。
YES	NO	根据 ADC 构建字典。	构建字典。
NO	YES	保留现有字典。	除去现有字典。
NO	NO	不执行任何操作。	不执行任何操作。

LOAD 实用程序的文件类型修饰符

表 31. LOAD 实用程序的有效文件类型修饰符: 所有文件格式

修饰符	描述
anyorder	此修饰符应与 *cpu_parallelism* 参数一起使用。指定不需要保留源数据顺序，这将能极大地提高 SMP 系统的性能。如果 *cpu_parallelism* 的值为 1，那么将忽略此选项。如果 SAVECOUNT > 0，那么此选项不受支持。这是因为在一致点后进行崩溃恢复将要求按顺序装入数据。
generatedignore	此修饰符通知 LOAD 实用程序: 所有生成列的数据在数据文件中都存在，但是应该忽略这些数据。这将导致所有生成列值都由该实用程序生成。此修饰符不能与 `generatedmissing` 或 `generatedoverride` 修饰符一起使用。
generatedmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含生成列的任何数据（甚至不包含 NULL）。这将导致所有生成列值都由该实用程序生成。此修饰符不能与 `generatedignore` 或 `generatedoverride` 修饰符一起使用。

表 31. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述

generatedoverride	此修饰符指示 LOAD 实用程序接受用户为表中的所有生成列提供的数据（这与这些类型的列的正常规则相反）。从另一数据库系统中迁移数据，或者在将使用 ROLLFORWARD DATABASE 命令 RECOVER DROPPED TABLE 选项恢复的数据装入到表中时，此修饰符就很有用。当使用此修饰符时，将拒绝存在下列情况的任何行: 对于不可空的生成列，不包含任何数据或者只包含 NULL 数据（SQL3116W）。当使用此修饰符时，表将处于“设置完整性暂挂”状态。要使该表脱离“设置完整性暂挂”状态，而不验证用户提供的值，请在完成装入操作后发出以下命令:
	`SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED`
	要使该表脱离“设置完整性暂挂”状态并强制验证用户提供的值，请在完成装入操作后发出以下命令:
	`SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.`
	当指定了此修饰符，并且任何分区键、维键或分布键中有生成列时，LOAD 命令会自动将该修饰符转换为 `generatedignore` 并继续执行装入。这与重新生成所有生成列值具有相同的效果。
	此修饰符不能与 `generatedmissing` 或 `generatedignore` 修饰符一起使用。

identityignore	此修饰符通知 LOAD 实用程序: 标识列的数据在数据文件中已存在，但是应该忽略该数据。这将导致所有标识值都由该实用程序生成。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 `identitymissing` 或 `identityoverride` 修饰符一起使用。
identitymissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含标识列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。此修饰符不能与 `identityignore` 或 `identityoverride` 修饰符一起使用。
identityoverride	仅当要装入的表中存在定义为 GENERATED ALWAYS 的标识列时，才应使用此修饰符。此修饰符指示该实用程序接受这样一列的显式、非 NULL 数据（这与这些类型的标识列的正常规则相反）。从另一数据库系统中迁移数据，并且必须将表定义为 GENERATED ALWAYS 时，或者在将使用 ROLLFORWARD DATABASE 命令 DROPPED TABLE RECOVERY 选项恢复的数据装入到表中时，此修饰符很有用。当使用此修饰符时，将拒绝存在下列情况的任何行: 对于标识列，不包含任何数据或者只包含 NULL 数据（SQL3116W）。此修饰符不能与 `identitymissing` 或 `identityignore` 修饰符一起使用。当使用此选项时，LOAD 实用程序将不会尝试维护或验证表的标识列中值的唯一性。
indexfreespace=*x*	*x* 是一个 0 到 99（包括 0 和 99 在内）的整数。该值被解释为重建索引时每一索引页中要保留为可用空间的百分比。使用 INDEXING MODE INCREMENTAL 装入时将忽略此选项。可添加页面中的第一个条目，而不受限制; 将添加后续条目以维持可用空间百分比阈值。缺省值是执行 CREATE INDEX 时所使用的值。 此值优先于在 CREATE INDEX 语句中指定的 PCTFREE 值; indexfreespace 选项只影响索引叶子页。

表 31. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
lobsinfile	*lob-path* 指定包含 LOB 数据的文件所在的路径。ASC、DEL 或 IXF 装入输入文件中包含下列文件的名称: 具有 LOB 列中的 LOB 数据的文件。 不支持将此选项与 CURSOR 文件类型一起使用。 使用"lobsinfile"修饰符时, LOBS FROM 子句指定 LOB 文件所在的位置。LOBS FROM 子句将隐式激活 LOBSINFILE 行为。LOBS FROM 子句将装入数据时要从其中搜索 LOB 文件的路径列表传递给 LOAD 实用程序。 每个路径都至少包含这样一个文件: 该文件至少包含数据文件中的"Lob 位置说明符"(LLS)所指向的一个 LOB。对于存储在 LOB 文件路径中的文件, LLS 就是对这些文件中的 LOB 所在位置的字符串表示。LLS 的格式为 *filename.ext.nnn.mmm/*, 其中 *filename.ext* 是包含 LOB 的文件名, *nnn* 是文件中的 LOB 的偏移量(以字节计), *mmm* 是 LOB 的长度(以字节计)。例如, 如果 `db2exp.001.123.456/` 字符串存储在数据文件中, 那么 LOB 位于 db2exp.001 文件中偏移量为 123 的位置, 其长度为 456 字节。 要指示一个空 LOB, 应将大小输入为 -1。如果将大小指定为 0, 那么会将它视作长度为 0 的 LOB。对于长度为 -1 的 LOBS, 将忽略偏移量和文件名。例如, 空 LOB 的 LLS 可能是 db2exp.001.7.-1/。
noheader	跳过头验证代码(只适用于要装入到单一分区数据库分区组中的表中的装入操作)。 如果对位于单一分区数据库分区组中的表使用了缺省 MPP 装入(PARTITION_AND_LOAD 方式), 那么不期望文件具有头。因此, 不需要 noheader 修饰符。如果使用了 LOAD_ONLY 方式, 那么期望文件具有头。仅当您想使用一个没有头的文件来执行 LOAD_ONLY 操作时, 才需要使用 noheader 修饰符。
norowwarnings	抑制关于被拒绝行的所有警告。
pagefreespace=*x*	*x* 是一个 0 到 100(包括 0 和 100 在内)的整数。该值被解释为每一数据页中要保留为可用空间的百分比。如果指定的值因最小行大小(例如, 至少为 3000 字节长并且 *x* 值为 50 的一行)而变得无效, 那么会将该行放在新的一页上。如果指定的值为 100, 那么每一行都将位于新的一页上。一个表的 PCTFREE 值确定每页指定的可用空间量。如果尚未设置装入操作的 pagefreespace 值或者表的 PCTFREE 值, 那么实用程序在每页上将填充尽可能多的空间。pagefreespace 设置的值将覆盖为表指定的 PCTFREE 值。
rowchangetimestampignore	此修饰符通知 LOAD 实用程序: ROW CHANGE TIMESTAMP 列的数据在数据文件中已存在, 但是应该忽略该数据。这将导致该实用程序生成所有 ROW CHANGE TIMESTAMP。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。这意味着, 对于 GENERATED ALWAYS 列, 将不会拒绝任何行。此修饰符不能与 `rowchangetimestampmissing` 或 `rowchangetimestampoverride` 修饰符一起使用。
rowchangetimestampmissing	如果指定了此修饰符, 那么该实用程序假定输入数据文件不包含 ROW CHANGE TIMESTAMP 列的任何数据(甚至不包含 NULL), 因此将为每一行生成一个值。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。此修饰符不能与 `rowchangetimestampignore` 或 `rowchangetimestampoverride` 修饰符一起使用。

表 31. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
rowchangetimestampoverride	仅当要装入的表中存在定义为 GENERATED ALWAYS 的 ROW CHANGE TIMESTAMP 列时，才应使用此修饰符。此修饰符指示该实用程序接受这样一列的显式、非 NULL 数据（这与这些类型的 ROW CHANGE TIMESTAMP 列的正常规则相反）。从另一数据库系统中迁移数据，并且必须将表定义为 GENERATED ALWAYS 时，或者在将使用 ROLLFORWARD DATABASE 命令 DROPPED TABLE RECOVERY 选项恢复的数据装入到表中时，此修饰符很有用。当使用此修饰符时，将拒绝存在下列情况的任何行: 对于 ROW CHANGE TIMESTAMP 列，不包含任何数据或者只包含 NULL 数据（SQL3116W）。此修饰符不能与 rowchangetimestampmissing 或 rowchangetimestampignore 修饰符一起使用。当使用此选项时，LOAD 实用程序将不会尝试维护或验证表的 ROW CHANGE TIMESTAMP 列中值的唯一性。

seclabelchar	指示输入源文件中的安全标号采用安全标号值的字符串格式，而不是采用缺省编码数字格式。当装入每个安全标号时，LOAD 实用程序会将它们转换为内部格式。如果字符串的格式不正确，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3242W）。如果字符串并不表示作为用于保护表的安全策略的一部分的有效安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3243W）。
	
	如果指定了 seclabelname 修饰符，那么不能指定此修饰符，否则装入将失败并且会返回错误（SQLCODE SQL3525N）。
	
	如果一个表中仅包含单个 DB2SECURITYLABEL 列，那么数据文件看起来可能类似如下:
	`"CONFIDENTIAL:ALPHA:G2"`
	`"CONFIDENTIAL;SIGMA:G2"`
	`"TOP SECRET:ALPHA:G2"`
	
	要装入或导入此数据，必须使用 SECLABELCHAR 文件类型修饰符:
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1`

seclabelname	指示输入源文件中的安全标号是由它们的名称指示的，而不是由缺省编码数字格式指示的。如果存在名称，那么 LOAD 实用程序会将它转换为适当的安全标号。如果不存在具有所指示的（用于保护表的）安全策略名称的安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3244W）。
	
	如果指定了 seclabelchar 修饰符，那么不能指定此修饰符，否则装入将失败并且会返回错误（SQLCODE SQL3525N）。
	
	如果一个表中仅包含单个 DB2SECURITYLABEL 列，那么数据文件可能包含与以下内容相似的安全标号名称:
	`"LABEL1"`
	`"LABEL1"`
	`"LABEL2"`
	
	要装入或导入此数据，必须使用文件类型修饰符 SECLABELNAME:
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1`
	
	注: 如果文件类型是 ASC，那么安全标号名称后面的任何空格都将被解释为该名称的一部分。为了避免此问题，可以使用文件类型修饰符 striptblanks 来确保除去空格。

表 31. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
totalfreespace=*x*	*x* 是一个大于或等于 0 的整数。该值被解释为: 表中的要追加至表末尾作为可用空间的总页数所占的百分比。例如, 如果 *x* 为 20, 而在装入数据后表中具有 100 个数据页, 那么将另外追加 20 个空的数据页。表中的数据页总数就将变为 120。数据页总数并不是表中的索引页数的一部分。此选项不会影响索引对象。如果在指定此选项的情况下完成了两次装入, 那么第二次装入将不会复用由第一次装入追加至末尾的额外空间。
usedefaults	如果已经指定了目标表列的源列, 但是源列中不包含一个或多个行实例的数据, 那么将装入缺省值。缺少的数据的示例如下: • 对于 DEL 文件: 为列值指定了两个相邻的列定界符 (",,") 或者两个相邻的列定界符之间还有任意数目的空格 (", ,")。 • 对于 DEL/ASC/WSF 文件: 没有足够的列数或者对于原始规范来说不是足够长的行。对于 ASC 文件, 并不将列值为 NULL 认为是显式丢失, 也不会用缺省值来代替 NULL 列值。NULL 列值是由数字、日期、时间和时间戳记列的所有空格字符来表示的, 或者是通过对任何类型的列使用 NULL INDICATOR 来指示该列为 NULL 这样来表示的。 在不使用此选项的情况下, 如果源列中不包含行实例的数据, 那么将发生下列情况之一: • 对于 DEL/ASC/WSF 文件: 如果该列可为空, 那么将装入 NULL。如果该列不可空, 那么实用程序将拒绝该行。

表 32. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式 (ASC/DEL)

修饰符	描述
codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输入数据集中的数据的代码页。在执行装入操作期间, 将字符数据 (以及在字符中指定的数字数据) 从此代码页转换为数据库代码页。 下列规则适用: • 对于纯 DBCS (图形)、混合 DBCS 和 EUC 来说, 定界符的范围是 x00 到 x3F。 • 对于使用 EBCDIC 代码页指定的 DEL 数据来说, 定界符可能与 shift-in 和 shift-out DBCS 字符不一致。 • nullindchar 必须指定标准 ASCII 代码集中包含的代码点 x20 与 x7F 之间 (包括 x20 和 x7F 在内) 的符号。这指的是 ASCII 符号和代码点。EBCDIC 数据可以使用相应的符号, 尽管代码点将不同。 不支持将此选项与 CURSOR 文件类型一起使用。
dateformat="*x*"	*x* 是源文件中的数据所采用的格式[1]。有效日期元素包括: YYYY - 年份 (四位数, 范围是 0000 到 9999) M - 月份 (一位数或两位数, 范围是 1 到 12) MM - 月份 (两位数, 范围是 1 到 12; 与 M 元素互斥) D - 日 (一位数或两位数, 范围是 1 到 31) DD - 日 (两位数, 范围是 1 到 31; 与 D 元素互斥) DDD - 一年中的某日 (三位数, 范围是 001 到 366; 与其他日或月份元素互斥) 对于未指定的每个元素, 将为它指定缺省值 1。以下是日期格式的一些示例: "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"

表 32. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述

dumpfile = *x*	*x* 是要将被拒绝行写入的异常文件的标准名称（按照服务器数据库分区）。每条记录最多写入 32 KB 数据。以下是一个示例，说明如何指定转储文件:
	`db2 load from data of del`
	`modified by dumpfile = /u/user/filename`
	`insert into table_name`
	将创建该文件，并归实例所有者所有。要覆盖缺省文件许可权，使用文件类型修饰符 dumpfileaccessall。
	注:
	1. 在分区数据库环境中，路径相对于正在装入的数据库分区来说应该是本地的，以便同时运行的装入操作不会尝试写入同一文件中。
	2. 文件内容以异步缓存方式写入磁盘中。对于已失败或已中断的装入操作，在执行 LOAD RESTART 后，无法明确知道已落实到磁盘的记录数，也不能保证一致性。对于一次性启动并完成的装入操作，只能假定文件是完整的。
	3. 如果指定的文件已存在，那么将不会重建该文件，但是将追加该文件。

dumpfileaccessall	当创建了转储文件时，为“OTHERS”授予读访问权。 此文件类型修饰符仅在下列情况下有效: 1. 将它与 dumpfile 文件类型修饰符一起使用 2. 用户对装入目标表具有 SELECT 特权 3. 它是在 UNIX 操作系统上的 DB2 服务器数据库分区上发出的 如果指定的文件已存在，那么将不会更改其许可权。
fastparse	使用时务必小心谨慎。减少对用户提供的列值进行语法检查的次数，以增强性能。确保表在结构上正确（该实用程序将执行足够的数据检查以防止分段违例或陷阱），但是，将不验证数据的一致性。只有您确信数据一致且正确时，才使用此选项。例如，如果用户提供的数据包含无效的时间戳记列值 :1>0-00-20-07.11.12.000000，那么在指定了 FASTPARSE 时，会将此值插入表，否则将拒绝此值。
implieddecimal	隐含的小数点所在的位置由列定义来确定；不再假定它位于值的末尾。例如，会将值 12345 作为 123.45 而不是 12345.00 装入 DECIMAL(8,2) 列。 此修饰符不能与 packeddecimal 修饰符一起使用。

timeformat="*x*"	*x* 是源文件中的时间格式[1]。有效时间元素包括:
	`H` - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24）
	`HH` - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥）
	`M` - 分钟（一位数或两位数，范围是 0 到 59）
	`MM` - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥）
	`S` - 秒（一位数或两位数，范围是 0 到 59）
	`SS` - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥）
	`SSSSS` - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥）
	`TT` - 正午指示符（AM 或 PM）
	对于未指定的每个元素，将为它指定缺省值 0。以下是时间格式的一些示例:
	`"HH:MM:SS"`
	`"HH.MM TT"`
	`"SSSSS"`

表 32. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
timestampformat="*x*"	*x* 是源文件中的时间戳记格式[1]。有效时间戳记元素包括: YYYY - 年份（四位数，范围是 0000 到 9999） M - 月份（一位数或两位数，范围是 1 到 12） MM - 月份（两位数，范围是 01 到 12；与 M 和 MMM 元素互斥） MMM - 月份（由三个不区分大小写的字母组成的月份名称缩写；与 M 和 MM 元素互斥） D - 日（一位数或两位数，范围是 1 到 31） DD - 日（两位数，范围是 1 到 31；与 D 元素互斥） DDD - 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥） H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24） HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥） M - 分钟（一位数或两位数，范围是 0 到 59） MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥） S - 秒（一位数或两位数，范围是 0 到 59） SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥） SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥） UUUUUU - 微秒（六位数，范围是 000000 到 999999；与所有其他微秒元素互斥） UUUUU - 微秒（五位数，范围是 00000 到 99999，映射至范围 000000 到 999990；与所有其他微秒元素互斥） UUUU - 微秒（四位数，范围是 0000 到 9999，映射至范围 000000 到 999900；与所有其他微秒元素互斥） UUU - 微秒（三位数，范围是 000 到 999，映射至范围 000000 到 999000；与所有其他微秒元素互斥） UU - 微秒（两位数，范围是 00 到 99，映射至范围 000000 到 990000；与所有其他微秒元素互斥） U - 微秒（一位数，范围是 0 到 9，映射至范围 000000 到 900000；与所有其他微秒元素互斥） TT - 正午指示符（AM 或 PM）

表 32. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述

timestampformat="x"（续）	对于未指定的 YYYY、M、MM、D、DD 或 DDD 元素，将为它们指定缺省值 1。对于未指定的 MMM 元素，将为它指定缺省值“Jan”。对于所有其他未指定的元素，将为它们指定缺省值 0。以下是一个表示时间戳记格式的示例:
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	MMM 元素的有效值包括: 'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov' 和 'dec'。这些值都不区分大小写。
	如果未指定 TIMESTAMPFORMAT 修饰符，那么 LOAD 实用程序将使用两种可能格式之一对时间戳记字段进行格式化:
	`YYYY-MM-DD-HH.MM.SS`
	`YYYY-MM-DD HH:MM:SS`
	LOAD 实用程序通过查看 DD 和 HH 之间的分隔符来选择格式。如果它是破折号“-”，那么 LOAD 实用程序将使用规则的破折号和点格式（YYYY-MM-DD-HH.MM.SS）。如果它是空格，那么 LOAD 实用程序需要使用冒号“:”来分隔 HH、MM 和 SS。
	在任一格式中，如果您包括微秒字段（UUUUUU），那么 load 实用程序将需要使用点“.”作为分隔符。YYYY-MM-DD-HH.MM.SS.UUUUUU 或 YYYY-MM-DD HH:MM:SS.UUUUUU 都是可接受的。
	以下示例说明如何将包含用户定义的日期和时间格式的数据装入到一个称为 schedule 的表中:
	`db2 load from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

usegraphiccodepage	如果给定了 usegraphiccodepage，那么假定装入到图形或双字节字符大对象（DBCLOB）数据字段的数据采用的是图形代码页。而假定其他数据采用的是字符代码页。图形代码页与字符代码页是相关联的。如果指定了 codepage 修饰符，那么 LOAD 实用程序通过该修饰符来确定字符代码页；如果未指定该修饰符，那么 LOAD 实用程序通过数据库的代码页来确定字符代码页。 仅当要恢复的表具有图形数据时，才应将此修饰符与由删除表恢复生成的定界数据文件一起使用。 **限制** 不能对 EXPORT 实用程序创建的 DEL 文件指定 usegraphiccodepage 修饰符，这是因为这些文件中包含只使用一种代码页编码的数据。文件中的双字节字符大对象（DBCLOB）也将忽略 usegraphiccodepage 修饰符。
xmlchar	指定采用字符代码页来 XML 文档进行编码。 处理采用指定的字符代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与字符代码页相匹配，否则将拒绝包含该文档的行。注意，字符代码页就是由文件类型修饰符 codepage 指定的值；如果未指定该修饰符，那么字符代码页就是应用程序代码页。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。

表 32. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
xmlgraphic	指定采用指定的图形代码页来对 XML 文档进行编码。 处理采用特定的图形代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与图形代码页相匹配，否则将拒绝包含该文档的行。注意，图形代码页就是由文件类型修饰符 codepage 指定的值的图形组件；如果未指定该修饰符，那么图形代码页就是应用程序代码页的图形组件。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。

表 33. LOAD 实用程序的有效文件类型修饰符: ASC 文件格式（非定界 ASCII）

修饰符	描述
binarynumerics	数字（但不是 DECIMAL）数据必须采用二进制格式，而不能采用字符表示。这样就避免了执行成本很高的转换。 只有使用由 reclen 选项指定的定长记录的位置 ASC 才支持此选项。 下列规则适用: • 除了 BIGINT、INTEGER 和 SMALLINT 之外，其他数据类型之间都不执行转换。 • 数据长度必须与它们的目标列定义相匹配。 • FLOAT 数据必须采用 IEEE 浮点格式。 • 无论装入操作在哪个平台上运行，都认为装入源文件中的二进制数据采用大尾数法。 在此修饰符影响的列数据中不能存在 NULL。当使用此修饰符时，空白将被解释为二进制值（但通常将解释为 NULL）。
nochecklengths	如果指定了 nochecklengths，那么即使源数据的列定义超过了目标表列大小，也会尝试装入每一行。如果代码页转换导致源数据缩小，那么可以成功地装入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。
nullindchar=*x*	*x* 是单个字符。将表示空值的字符更改为 *x*。*x* 的缺省值为 Y[2]。 对于 EBCDIC 数据文件，除非字符是英文字母，否则此修饰符将区分大小写。例如，如果将空指示符指定为字母 N，那么 n 也会被识别为空指示符。

表 33. LOAD 实用程序的有效文件类型修饰符: ASC 文件格式（非定界 ASCII）(续)

修饰符	描述

packeddecimal	直接装入压缩十进制数据，这是因为 binarynumerics 修饰符不包括 DECIMAL 字段类型。
	
	只有使用由 reclen 选项指定的定长记录的位置 ASC 才支持此选项。
	
	带符号的半字节的支持值包括:
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	在此修饰符影响的列数据中不能存在 NULL。当使用此修饰符时，空白将被解释为二进制值（但通常将解释为 NULL）。
	
	无论在哪个服务器平台上，都认为装入源文件中的二进制数据的字节顺序是大尾数法；也就是说，当在 Windows 操作系统上使用此修饰符时，不能保留字节顺序。
	
	此修饰符不能与 implieddecimal 修饰符一起使用。

reclen=*x*	*x* 是一个整数，最大值为 32 767。会读取每行的 *x* 个字符，但未使用换行符来指示行的末尾。
striptblanks	当将数据装入到一个变长字段时，将截断任何尾部空格。如果未指定此选项，那么将保留空格。 不能将此选项与 striptnulls 同时指定。它们是互斥选项。此选项将替换过时的 t 选项，支持此过时选项只是为了保持与早前版本的兼容性。
striptnulls	当将数据装入到一个变长字段时，将截断任何尾部 NULL（0x00 字符）。如果未指定此选项，那么将保留 NULL。 不能将此选项与 striptblanks 同时指定。它们是互斥选项。此选项将替换过时的 padwithzero 选项，支持此过时选项只是为了保持与早前版本的兼容性。

zoneddecimal	装入分区十进制数据，这是因为 BINARYNUMERICS 修饰符不包括 DECIMAL 字段类型。只有使用由 RECLEN 选项指定的定长记录的位置 ASC 才支持此选项。
	
	带符号的半字节值可以是下列其中一项:
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	受支持的位数值是 0x0 到 0x9。
	
	受支持的区域值是 0x3 和 0xF。

表 34. LOAD 实用程序的有效文件类型修饰符: DEL 文件格式（定界 ASCII）

修饰符	描述

chardel*x*	*x* 是单个字符串定界符。缺省值是双引号（"）。使用指定的字符而不是使用双引号将字符串引起来[23]。如果您想显式地指定双引号（"）作为字符串定界符，那么应按如下所示指定双引号:
	`modified by chardel""`
	
	也可以指定单引号（'）作为字符串定界符，如下所示:
	`modified by chardel''`

coldel*x*	*x* 是一个单字符列定界符。缺省值是逗号（,）。使用指定字符而不是逗号来表示列的末尾[23]。

表 34. LOAD 实用程序的有效文件类型修饰符: DEL 文件格式（定界 ASCII）(续)

修饰符	描述
decplusblank	加号字符。导致在正的十进制值前面加上空格而不是加号（+）。缺省操作是在正的十进制值前面加上加号。
decpt*x*	*x* 是单个字符，它取代句点作为小数点字符。缺省值是句点（.）。使用指定字符而不是句点作为小数点字符[23]。

delprioritychar	当前，定界符的缺省优先级为记录定界符、字符定界符和列定界符。此修饰符通过将定界符优先级还原为字符定界符、记录定界符和列定界符，来保护依赖于旧的优先级的现有应用程序。语法:
	`db2 load ... modified by delprioritychar ...`
	例如，用以下 DEL 数据文件作为示例:
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	如果指定了 delprioritychar 修饰符，那么此数据文件中将只有两行。第二个 <row delimiter> 将被解释为第二行的第一个数据列，而第一个和第三个 <row delimiter> 被解释为实际的记录定界符。如果未指定此修饰符，那么此数据文件中将有三行，每一行都用 <row delimiter> 定界。

keepblanks	保留类型为 CHAR、VARCHAR、LONG VARCHAR 或 CLOB 的每个字段中的前导空格和尾部空格。如果未指定此选项，那么会除去字符定界符外部的所有前导空格和尾部空格，并且会在表中插入 NULL 表示所有空白字段。
	以下示例说明如何将数据装入到称为 TABLE1 的表中，同时还要保留数据文件中的所有前导空格和尾部空格:
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nochardel	LOAD 实用程序将假定在列定界符之间找到的所有字节都是列数据的一部分。字符定界符将被解析为列数据的一部分。如果数据是使用 DB2 导出的，那么不应指定此选项（除非在导出时指定了 nochardel）。提供此修饰符的目的是支持不具有字符定界符的供应商数据文件。未正确使用此修饰符可能会导致数据丢失或毁坏。 不能将此选项与 chardelx、delprioritychar 或 nodoubledel 同时指定。它们是互斥选项。
nodoubledel	不识别双字符定界符。

表 35. LOAD 实用程序的有效文件类型修饰符: IXF 文件格式

修饰符	描述
forcein	指示实用程序接受数据（即使代码页不匹配也接受），并且阻止代码页之间进行转换。 将检查定长目标字段，以验证它们对于数据来说是否足够大。如果指定了 nochecklengths，那么不会进行检查，并且将尝试装入每一行。
nochecklengths	如果指定了 nochecklengths，那么即使源数据的列定义超过了目标表列大小，也会尝试装入每一行。如果代码页转换导致源数据缩小，那么可以成功地装入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。

注:

1. 日期格式字符串两边必须具有双引号。字段分隔符不能包含下列任何字符: a-z、A-Z 和 0-9。字段分隔符不应与 DEL 文件格式中的字符定界符或字段定界符相同。如果元素的开始和结束位置是明确的，那么字段分隔符是可选的。如果使用了诸如 D、H、M 或 S 之类的元素（取决于修饰符），那么由于条目长度是可变的，因此可能存在不明确性。

   对于时间戳记格式，必须要注意避免月份描述符与分钟描述符之间的不明确性，这是因为它们都使用字母 M。月份字段必须与其他日期字段相邻。而分钟字段必须与其他时间字段相邻。以下是一些不明确的时间戳记格式:

   ```
   "M"（既可能是月份，也可能是分钟）
      "M:M"（无法区分哪个是月份，哪个是分钟）
      "M:YYYY:M"（两者都将被解释为月份。）
      "S:M:YYYY"（与时间值和日期值都相邻）
   ```

   在不明确的情况下，实用程序将报告一条错误消息，并且操作将失败。

   以下是一些明确的时间戳记格式:

   ```
   "M:YYYY"（表示月份）
      "S:M"（表示分钟）
      "M:YYYY:S:M"（前者表示月份，后者表示分钟）
      "M:H:YYYY:M:D"（前者表示分钟，后者表示月份）
   ```

   在某些字符（例如，双引号和反斜杠）前面必须添加转义字符（例如，@2329。

2. 为 chardel、coldel 或 decpt 文件类型修饰符指定的字符值必须采用源数据的代码页来指定。

   可以使用 xJJ 或 0xJJ 语法来指定字符代码点（而不是字符符号）。其中 JJ 是代码点的十六进制表示法。例如，要指定 # 字符作为列定界符，可使用下列方法之一:

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

3. *移动数据时的定界符注意事项*列示了可以用作定界符的字符存在的限制。
4. 如果您尝试将不受支持的文件类型与 MODIFIED BY 选项配合使用，那么 LOAD 实用程序将不会发出警告。如果尝试这样做，那么装入操作将失败，并且会返回错误代码。
5. 导入到包含隐式隐藏的 ROW CHANGE TIMESTAMP 列的表中时，将不采用该列的隐式隐藏属性。因此，如果列的数据在要导入的数据中不存在，且不存在任何显式列列表，那么*必须*在导入命令中指定 rowchangetimestampmissing 文件类型修饰符。

表 36. 同时使用 codepage 和 usegraphiccodepage 时的 LOAD 行为

codepage=N	usegraphiccodepage	LOAD 行为
缺少	缺少	假定文件中的所有数据都采用数据库代码页，而不采用应用程序代码页，即使指定了 CLIENT 选项亦如此。
存在	缺少	假定文件中的所有数据都采用代码页 N。 **警告**: 如果 N 是单字节代码页，那么将图形数据装入到数据库中时将毁坏该数据。

表 36. 同时使用 codepage 和 usegraphiccodepage 时的 LOAD 行为 (续)

codepage=N	usegraphiccodepage	LOAD 行为
缺少	存在	假定文件中的字符数据采用数据库代码页，即使指定了 CLIENT 选项亦如此。假定图形数据要采用数据库图形数据的代码页，即使指定了 CLIENT 选项亦如此。 如果数据库代码页是单字节，那么假定所有数据都采用数据库代码页。 **警告**: 将图形数据装入到单字节数据库中时将毁坏该数据。
存在	存在	假定字符数据采用代码页 N。假定图形数据采用图形代码页 N。 如果 N 是单字节或双字节代码页，那么假定所有数据都采用代码页 N。 **警告**: 如果 N 是单字节代码页，那么将图形数据装入到数据库中时将毁坏该数据。

使用 ADMIN_CMD 过程的 LOAD 命令

将数据装入 DB2 表中。服务器上的数据以文件、磁带或命名管道的形式存在。如果表的 COMPRESS 属性设置为 YES，那么装入的数据将受到每个数据的压缩情况以及在表中已存在字典的数据库分区的支配。

指向 第 227 页的『LOAD 实用程序的文件类型修饰符』的快速链接。

限制

LOAD 实用程序不支持在层次结构级别装入数据。LOAD 实用程序与范围集群表不兼容。

作用域

可通过单个请求向多个数据库分区发出此命令。

权限

为下列其中一项:

- *sysadm*
- *dbadm*
- 对数据库的 LOAD 权限，以及下列特权
  - 以 INSERT 方式、TERMINATE 方式（用于终止先前的装入插入操作）或 RESTART 方式（用于重新启动先前的装入插入操作）调用 LOAD 实用程序时，对表的 INSERT 特权
  - 以 REPLACE 方式、TERMINATE 方式（用于终止先前的装入替换操作）或 RESTART 方式（用于重新启动先前的装入替换操作）调用 LOAD 实用程序时，对表的 INSERT 和 DELETE 特权
  - 作为装入操作使用异常表时，对异常表的 INSERT 特权

- 要将数据装入到包含受保护列的表中，会话授权标识必须拥有允许对该表中所有受保护列执行写访问的 LBAC 凭证。否则，装入将失败并且返回错误（SQLSTATE 5U014）。
- 要将数据装入包含受保护行的表中，会话授权标识必须拥有满足下列条件的安全标号:
  - 它是用于保护表的安全策略的一部分
  - 已将它授予会话授权标识以进行写访问或所有访问

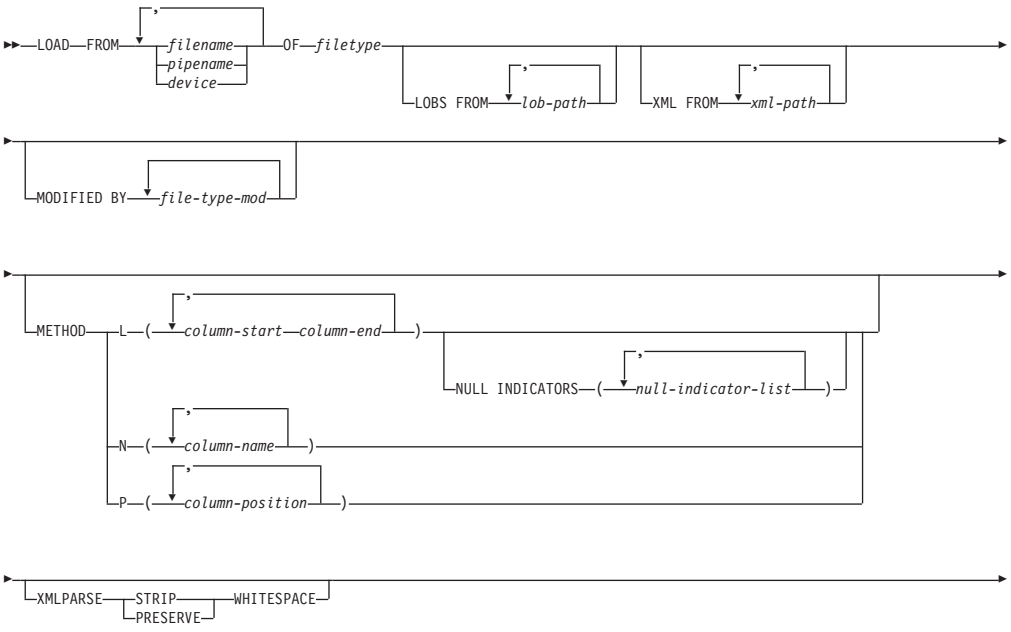
  如果会话授权标识没有这样的安全标号，那么装入将失败并且返回错误（SQLSTATE 5U014）。如果会话授权标识的 LBAC 凭证不允许写入用于保护数据中的已装入行的安全标号，那么此安全标号将用来保护该行。但是，如果用于保护该表的安全策略是使用 CREATE SECURITY POLICY 语句的 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项来创建的，那么不会发生这种情况。在这种情况下，装入将失败并且返回错误（SQLSTATE 42519）。
- 如果指定了 REPLACE 选项，那么会话授权标识必须有权删除该表。

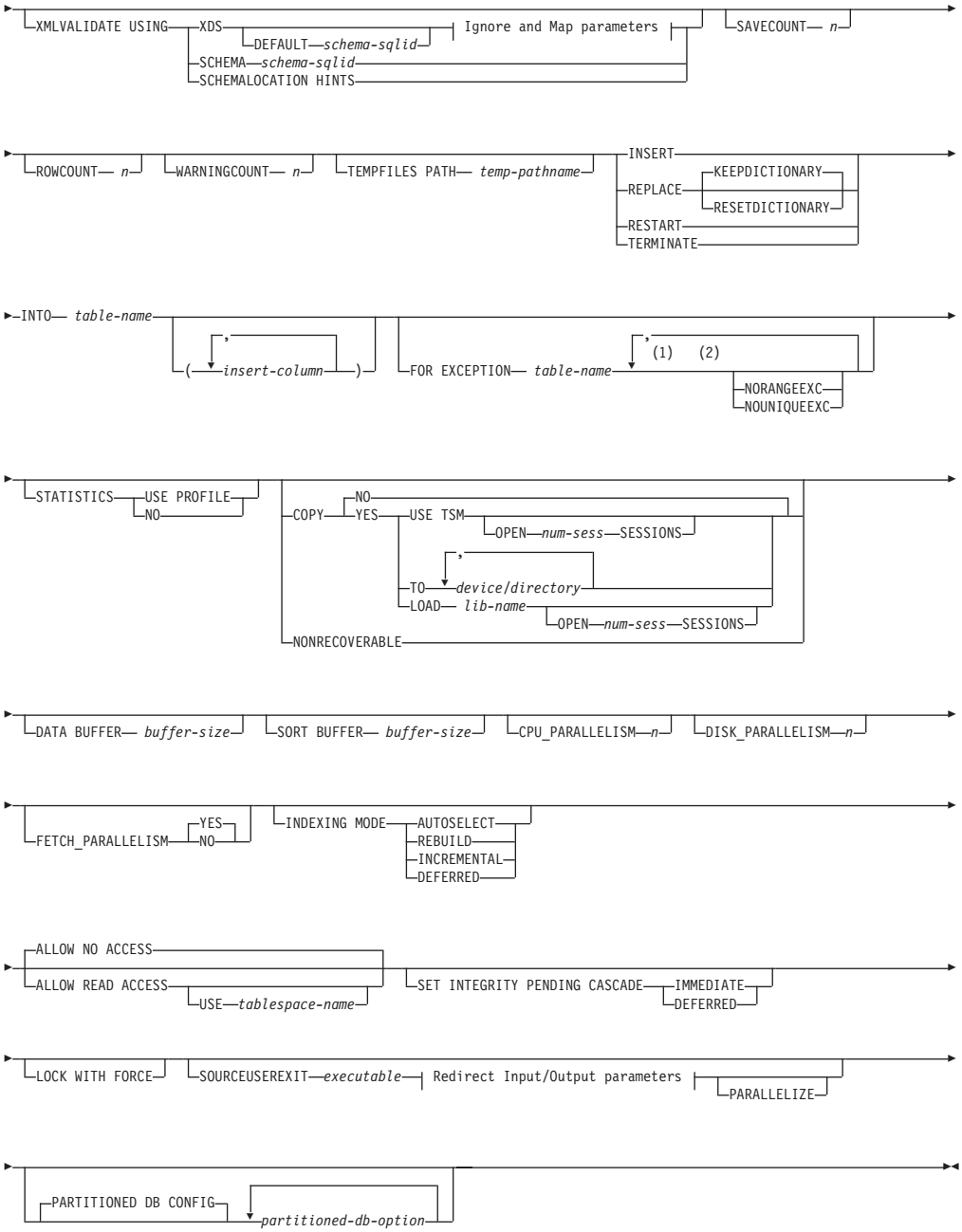
由于所有装入进程（通常还包括所有 DB2 服务器进程）都由实例所有者拥有，并且所有这些进程都使用实例所有者的标识来访问所需的文件，因此，实例所有者必须对输入数据文件具有读访问权。无论谁调用该命令，实例所有者都必须能够读取这些输入数据文件。

必需的连接

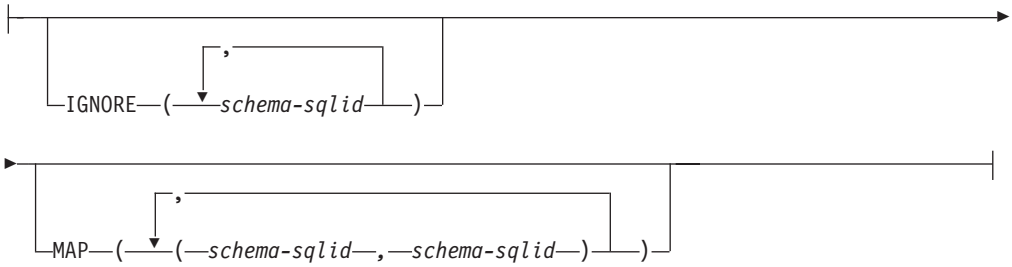
实例。不需要显式连接。如果已经与数据库建立了连接，那么会尝试与本地实例进行隐式连接。

命令语法

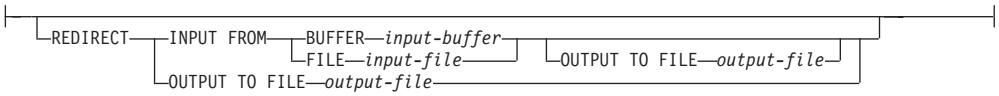




Ignore and Map parameters:



Redirect Input/Output parameters:



注:

1 这些关键字可以按任意顺序出现。

2 这些关键字中的每一个都只能出现一次。

命令参数

FROM *filename* | *pipename* | *device*

注:

1. 如果通过*使用 ADMIN_CMD 过程的 EXPORT 命令*将数据导出到文件，那么数据文件归受防护用户标识所有。实例所有者通常不能访问此文件。要从 CLP 或 ADMIN_CMD 过程运行 LOAD 命令，实例所有者标识必须能够访问数据文件，以保证将对数据文件的读访问权授予实例所有者。
2. 如果多个 IXF 文件在物理位置上是分隔开的，但在逻辑上是一个文件，那么支持装入这些文件中的数据。如果这些文件在逻辑上和物理位置上都是分开的，那么不支持装入这些文件中的数据。（如果多个物理文件都是通过调用一次 EXPORT 命令来创建的，那么在逻辑上会将这些文件视为是一个文件。）

OF *filetype*

指定数据的格式:

- ASC（非定界 ASCII 格式）。
- DEL（定界 ASCII 格式）。
- IXF（PC 版本的集成交换格式）是一种专门供 DB2 使用的二进制格式。
- CURSOR（对 SELECT 或 VALUES 语句声明的游标）。

LOBS FROM *lob-path*

包含要装入的 LOB 值的数据文件的路径。该路径必须以斜杠（/）结尾。LOB 数据文件的名称存储在将装入到 LOB 列的那列的主数据文件（ASC、DEL 或 IXF）中。最多可指定 999 个路径。这将隐式激活 LOBSINFILE 行为。

当将此选项与 CURSOR 文件类型一起指定时，将忽略此选项。

MODIFIED BY *file-type-mod*

指定文件类型修饰符选项。请参阅 第 227 页的『LOAD 实用程序的文件类型修饰符』。

METHOD

L 指定要将数据装入到的列的起始列号和结束列号。列号就是与一个数据行的开头的字节偏移量。它是从 1 开始计数的。此方法只能用于 ASC 文件，并且对于该文件类型是唯一有效的方法。

NULL INDICATORS *null-indicator-list*

仅当指定了 METHOD L 参数时（即，输入文件是 ASC 文件），才能使用此选项。空指示符列表是用于指定每个空指示

符字段的正整数的列表，各个正整数之间用逗号隔开。列号是空指示符字段与一个数据行的开头之间的字节偏移量。对于 METHOD L 参数中定义的每个数据字段，在空指示符列表中必须有一个其对应的条目。如果列号为 0，那么表示相应的数据字段中始终包含数据。

如果空指示符列中的值为 Y，那么表示列数据为 NULL。如果空指示符列中的值是除了 Y 之外的任何字符，那么表示列数据不是 NULL，并且将装入由 METHOD L 选项指定的列数据。

可以使用 MODIFIED BY 选项来更改空指示符。

N 指定要装入的数据文件中的列名。这些列名的大小写必须与系统目录中相应名称的大小写相匹配。不可空的每个表列在 METHOD N 列表中都应该具有相应的条目。例如，给定数据字段 F1、F2、F3、F4、F5 和 F6，以及表列 C1 INT、C2 INT NOT NULL、C3 INT NOT NULL 和 C4 INT，那么 `method N (F2, F1, F4, F3)` 是有效请求，而 `method N (F2, F1)` 是无效请求。此方法只能用于文件类型 IXF 或 CURSOR。

P 指定要装入的输入数据字段的字段编号（从 1 开始计数）。不可空的每个表列在 METHOD P 列表中都应该具有相应的条目。例如，给定数据字段 F1、F2、F3、F4、F5 和 F6，以及表列 C1 INT、C2 INT NOT NULL、C3 INT NOT NULL 和 C4 INT，那么 `method P (2, 1, 4, 3)` 是有效请求，而 `method P (2, 1)` 是无效请求。此方法只能用于文件类型 IXF、DEL 或 CURSOR，并且对于 DEL 文件类型是唯一有效的方法。

XML FROM *xml-path*

指定一个或多个包含 XML 文件的路径。XDS 包含在主数据文件（ASC、DEL 或 IXF）中将装入到 XML 列中的那一列中。

XMLPARSE

指定如何解析 XML 文档。如果未指定此选项，那么将由 CURRENT XMLPARSE OPTION 专用寄存器的值来确定 XML 文档的解析行为。

STRIP WHITESPACE

指定在解析 XML 文档时要去掉空格。

PRESERVE WHITESPACE

指定在解析 XML 文档时不去掉空格。

XMLVALIDATE

指定在适当的情况下将针对某一模式来验证 XML 文档。

USING XDS

将针对由主数据文件中的 XML 数据说明符（XDS）标识的 XML 模式来验证 XML 文档。缺省情况下，如果使用 USING XDS 子句调用了 XMLVALIDATE 选项，那么将由 XDS 的 SCH 属性来确定用来执行验证的模式。如果 XDS 中不存在 SCH 属性，那么除非缺省模式是由 DEFAULT 子句指定的，否则将不会进行模式验证。

可以使用 DEFAULT、IGNORE 和 MAP 子句来修改模式确定行为。这三个可选子句直接应用于 XDS 的指定，但是它们不会互相应用。例如，如果由于 DEFAULT 子句指定了某一模式而选择了该模式，那么，

即使 IGNORE 子句也指定了该模式，该模式仍不会被忽略。类似，如果由于将某一模式指定为 MAP 子句对的第一部分而选择了该模式，那么，即使在另一个 MAP 子句对的第二部分中也指定了该模式，该模式仍不会被重新映射。

USING SCHEMA *schema-sqlid*
将针对具有指定的 SQL 标识的 XML 模式来验证 XML 文档。在这种情况下，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

USING SCHEMALOCATION HINTS
将针对由源 XML 文档中的 XML 模式位置提示标识的模式来验证 XML 文档。如果在 XML 文档中找不到 schemaLocation 属性，那么将不执行验证。指定 USING SCHEMALOCATION HINTS 子句时，将对所有 XML 列忽略 XML 数据说明符（XDS）的 SCH 属性。

请参阅下面的 XMLVALIDATE 选项示例。

IGNORE *schema-sqlid*
仅当指定了 USING XDS 参数时，才能使用此选项。如果 SCH 属性标识了一种或多种模式，那么 IGNORE 子句指定这些模式中要忽略的模式列表。如果 SCH 属性存在于已装入的 XML 文档的 XML 数据说明符中，并且由 SCH 属性标识的模式包含在要忽略（IGNORE）的模式列表中，那么将不会对已装入的 XML 文档进行模式验证。

注:

如果某一模式是在 IGNORE 子句中指定的，那么该模式不能存在于 MAP 子句中模式对的左边。

IGNORE 子句仅适用于 XDS。如果 IGNORE 子句指定了由 MAP 子句映射的模式，那么以后不会忽略该模式。

DEFAULT *schema-sqlid*
仅当指定了 USING XDS 参数时，才能使用此选项。通过 DEFAULT 子句指定的模式标识: 当已装入的 XML 文档的 XML 数据说明符（XDS）不包含用于标识 XML 模式的 SCH 属性时要用于验证的模式。

DEFAULT 子句优先于 IGNORE 和 MAP 子句。如果 XDS 满足 DEFAULT 子句，那么将忽略 IGNORE 和 MAP 规范。

MAP *schema-sqlid*
仅当指定了 USING XDS 参数时，才能使用此选项。使用 MAP 子句来指定要使用的替代模式，这些模式将取代由已装入的每个 XML 文档的 XML 数据说明符（XDS）的 SCH 属性所指定的那些模式。MAP 子句指定一个或多个模式对的列表，而每个模式对都表示一个模式与另一个模式之间的映射。模式对中的第一个模式表示 XDS 中的 SCH 属性引用的模式。模式对中的第二个模式表示应该用来执行模式验证的模式。

如果某一模式存在于 MAP 子句中的模式对的左边，那么不能在 IGNORE 子句中也指定该模式。

一旦应用了模式对映射，其结果就是最终结果。映射操作不是过渡操作，因此，以后不会将所选择的模式应用于另一个模式对映射。

不能多次映射同一模式，这就意味着该模式不会多次出现在一个模式对的左边。

SAVECOUNT *n*

指定 LOAD 实用程序在每隔 *n* 行之后就要建立一致点。此值被转换为页面计数，并且向上取整为扩展数据块大小的间隔。由于在每个一致点都会发出消息，因此，如果将使用 LOAD QUERY 来监视装入操作，那么应选择此选项。如果 *n* 的值不是足够大，那么在每个一致点执行的活动的同步将影响性能。

缺省值为 0，这意味着除非确实需要建立一致点，否则将不会建立任何一致点。

当将此选项与 CURSOR 文件类型一起指定时，将忽略此选项。

ROWCOUNT *n*

指定要装入的文件中的 *n* 个物理记录。允许用户只装入文件中前面的 *n* 行。

WARNINGCOUNT *n*

在发出 *n* 个警告后停止装入操作。如果希望不产生警告，那么设置此参数，但是需要验证使用的是正确的文件和表。如果未正确指定装入文件或目标表，那么 LOAD 实用程序将对它试图装入的每一行生成一个警告，这将导致装入失败。如果 *n* 为 0，或者未指定此选项，那么无论发出多少个警告，都将继续执行装入操作。如果因达到了警告数的阈值而停止了装入操作，那么可以采用 RESTART 方式开始执行另一个装入操作。装入操作将自动从最后一个一致点继续执行。或者，可以采用 REPLACE 方式启动另一个装入操作，并从输入文件开头开始装入。

TEMPFILES PATH *temp-pathname*

指定在执行装入操作期间创建临时文件时要使用的路径（相对于服务器数据库分区而言，此路径应该是标准路径）的名称。

临时文件将占用文件系统空间。有时，此空间需求是相当大的。以下是估计应该为所有临时文件分配的文件系统空间量:

- LOAD 实用程序生成的每条消息需要 136 字节。
- 如果数据文件包含长整型字段数据或 LOB，那么需要 15 KB 的开销。如果指定了 INSERT 选项，并且表中已经具有大量的长整型字段或 LOB 数据，那么需要的空间量会显著增加。

INSERT

执行 LOAD 实用程序时可以采用的四种方式之一。将已装入的数据添加至表，但不更改现有表数据。

REPLACE

执行 LOAD 实用程序时可以采用的四种方式之一。从表中删除现有的所有数据，然后插入已装入的数据。表定义和索引定义不会发生更改。如果在层次结构之间移动数据时使用了此选项，那么只能替换整个层次结构的数据，而不能替换单个子表的数据。

KEEPDICTIONARY

在执行 LOAD REPLACE 操作期间将保留现有压缩字典。如果表 COMPRESS 属性是 YES，那么将使用执行装入调用前存在的字典来压缩新替换的数据。如果在此之前，表中不存在任何字典，那么只要表 COMPRESS 属性是 YES，就会使用要替换到表中的数据来构建新字典。在这种情况下，构建压缩字典所需的数据量受 ADC 策略的约束。此数据将以未压缩的形式填充到表中。将该字典插入表中后，余下要

装入的数据将使用此字典进行压缩。这是缺省参数。请参阅下面的表 1 以获取相关摘要。

表 37. LOAD REPLACE KEEPDICTIONARY

压缩	存在字典	结果
Y	Y	保留字典；所有输入行都将使用现有字典进行压缩。
Y	N	只有存在足够的用户数据时，才将新字典插入表中；在构建字典后，余下的行将使用该字典进行压缩。
N	Y	保留字典；不压缩所有输入行。
N	N	无影响；不压缩所有行。

RESETDICTIONARY

如果表 COMPRESS 属性是 YES，那么此伪指令指示 LOAD REPLACE 处理操作为表数据对象构建新字典。如果 COMPRESS 属性是 NO，而表中已存在字典，那么会将该字典除去并且不会将任何新字典插入该表中。可以构建仅包含一条用户记录的压缩字典。如果装入的数据集大小为 0 且预先存在字典，那么将不保留该字典。使用此伪指令构建字典所需的数据量不受 ADC 策略的约束。请参阅下面的表 2 以获取相关摘要。

表 38. LOAD REPLACE RESETDICTIONARY

压缩	存在字典	结果
Y	Y	构建新字典*；构建字典后，将使用该字典压缩余下要装入的行。
Y	N	构建新字典；构建字典后，将使用该字典压缩余下的行。
N	Y	除去字典；不压缩所有输入行。
N	N	无影响；不压缩所有行。

* 如果存在字典且启用了压缩属性，但没有任何记录可供装入到表分区中，那么无法构建新字典且 RESETDICTIONARY 操作将不保留现有字典。

TERMINATE

执行 LOAD 实用程序时可以采用的四种方式之一。终止先前已中断的装入操作，并将该操作回滚到开始执行它时的时间点，即使已经超过了一致点亦如此。该操作中涉及到的任何表空间的状态将恢复为正常状态，并且会使所有表对象保持一致（索引对象可能会被标记为无效。在这种情况下，下一次进行访问时将自动重建索引）。如果终止的装入操作是 LOAD REPLACE，那么在执行 LOAD TERMINATE 操作后，表将被截断为一个空表。如果终止的装入操作是 LOAD INSERT，那么在执行 LOAD TERMINATE 操作后，表将保留它的所有原始记录。请参阅下面的表 3 以获取字典管理的摘要。

LOAD TERMINATE 选项不会使表空间脱离“备份暂挂”状态。

RESTART

执行 LOAD 实用程序时可以采用的四种方式之一。重新启动先前已中断的装入操作。装入操作将自动从装入、构建或删除阶段中的最后一个一致点继续执行。请参阅下面的表 4 以获取字典管理的摘要。

INTO *table-name*

指定要将数据装入到的数据库表。此表不能是系统表或者已声明临时表。可指

定别名，或者指定标准表名或非限定表名。标准表名的格式为 schema.tablename。如果指定了非限定表名，那么将使用 CURRENT SCHEMA 语句来限定表。

insert-column

指定要将数据插入到的表列。

LOAD 实用程序将无法解析名称中包含一个或多个空格的那些列。例如，

将因 Int 4 列而失败。解决方案是将这些列名用双引号引起来:

FOR EXCEPTION *table-name*

指定要将错误行复制到的异常表。将复制违反唯一索引或主键索引的任何行。如果指定了非限定表名，那么将使用 CURRENT SCHEMA 语句来限定表。

不会将写入异常表的信息写入转储文件。在分区数据库环境中，必须为定义了装入表的那些数据库分区定义异常表。除此之外，转储文件将包含无法装入的行，这是因为这些行无效或者具有语法错误。

NORANGEEXC

指示如果某行因违反范围而被拒绝，那么不会将该行插入到异常表中。

NOUNIQUEEXC

指示如果某行因违反唯一约束而被拒绝，那么不会将该行插入到异常表中。

STATISTICS USE PROFILE

指示 LOAD 实用程序在装入操作期间根据为此表定义的概要文件来收集统计信息。必须在执行装入前创建此概要文件。此概要文件是使用 RUNSTATS 命令创建的。如果此概要文件不存在，但是您指示装入操作根据此概要文件来收集统计信息，那么会返回警告，并且不会收集统计信息。

STATISTICS NO

指定将不会收集统计信息，目录中的统计信息也不会改变。这是缺省值。

COPY NO

指定如果启用了正向恢复（即，打开了 *logretain* 或 *userexit*），那么表所在的表空间将被置于“备份暂挂”状态。COPY NO 选项还会将表空间状态变为“正在装入”表空间状态。这是一种瞬态状态，当装入操作完成或者异常中止时，此状态就会消失。在备份表空间或者备份整个数据库后，才能更新或删除表空间中的任何表所包含的数据。但是，可以使用 SELECT 语句来访问任何表中的数据。

对可恢复的数据库执行指定了 COPY NO 选项的 LOAD 命令时，会将表空间置于“备份暂挂”状态。例如，执行指定了 COPY NO 和 INDEXING MODE DEFERRED 选项的 LOAD 命令时，将需要刷新索引。对表的某些查询可能需要进行索引扫描，并且在刷新索引前将不会成功。如果索引位于一个处于“备份暂挂”状态的表空间中，那么无法刷新该索引。在这种情况下，在执行备份前将不允许访问表。当查询访问索引时，数据库将自动完成索引刷新。如果未指定 COPY NO、COPY YES 或 NONRECOVERABLE 之一，并且数据库是可恢复的（启用了 **logretain** 或 **logarchmeth1**），那么缺省值为 COPY NO。

COPY YES

指定将保存已装入数据的副本。如果禁用了正向恢复（即，关闭了 *logretain* 和 *userexit*），那么此选项无效。

USE TSM

指定将使用 Tivoli Storage Manager（TSM）来存储副本。

OPEN *num-sess* **SESSIONS**

将与 TSM 或供应商产品一起使用的 I/O 会话数。缺省值为 1。

TO *device/directory*

指定将创建复制映像的设备或目录。

LOAD *lib-name*

共享库（在 Windows 操作系统上为 DLL）的名称，该共享库包含要使用的供应商备份与复原 I/O 函数。也可以包含完整路径。如果未提供完整路径，将缺省为用户出口程序所在的路径。

NONRECOVERABLE

指定装入事务将标记为不可恢复，并且后续前滚操作不能将其恢复。Rollforward 实用程序将跳过该事务，并且会将装入数据的表标记为"无效"。该实用程序还将忽略该对表执行的任何后续事务。在完成前滚操作后，只能删除这样的表，或者从完成不可恢复的装入操作后的落实点后生成的备份（完整或表空间）来复原该表。

如果指定了此选项，在装入操作完成后就不会将表空间置于"备份暂挂"状态，并且在装入操作执行期间不必创建所装入数据的副本。如果未指定 COPY NO、COPY YES 或 NONRECOVERABLE 之一，并且数据不可恢复（未启用 **logretain** 或 **logarchmeth1**），那么缺省值为 NONRECOVERABLE。

WITHOUT PROMPTING

指定数据文件列表中包含要装入的所有文件，并且所列示的设备或目录足以满足整个装入操作的需求。如果找不到下一个输入文件，或者在完成装入操作前填充了复制目标，那么装入操作将失败，并且表将保持处于装入暂挂状态。

DATA BUFFER *buffer-size*

指定要用作用于传送实用程序中数据的缓存空间的 4 KB 页数（不考虑并行度）。如果指定值小于算术最小值，那么将使用需要的最少资源，但不会返回警告。

此内存是直接从实用程序堆中分配的，可通过数据库配置参数 *util_heap_sz* 来修改此内存大小。

如果未指定值，那么实用程序在运行时将计算智能缺省值。该缺省值取决于在实例化装入程序时实用程序堆中的可用空间以及表的某些特征。

SORT BUFFER *buffer-size*

此选项指定一个在装入操作期间将覆盖 SORTHEAP 数据库配置参数的值。仅当装入具有索引的表以及 INDEXING MODE 参数未指定为 DEFERRED 时，此选项才有效。为此选项指定的值不能超过 SORTHEAP 的值。此参数对于在不更改 SORTHEAP 值的情况下调整在装入具有许多索引的表时所使用的排序内存是很有用的，这还会影响常规查询处理。

CPU_PARALLELISM *n*

指定当构建表对象时，LOAD 实用程序为了解析、转换和格式化记录而创建的进程数或线程数。此参数旨在利用分区内并行性。当装入已预先排序的数据时，此参数特别有用，这是因为将保留源数据中的记录顺序。如果此参数的值为零或者尚未指定，那么 LOAD 实用程序在运行时将使用智能缺省值（它通常基于可用 CPU 的数目）。

注:

1. 如果对包含 LOB 或 LONG VARCHAR 字段的表使用此参数，那么无论系统 CPU 的数目或者用户指定的值是多少，此参数的值都将为 1。
2. 如果对 SAVECOUNT 参数指定一个较小的值，那么会导致装入程序执行许多 I/O 操作来同时清空数据和表元数据。当 CPU_PARALLELISM 大于 1 时，清仓操作是异步执行的，并且允许装入程序利用 CPU。当 CPU_PARALLELISM 设置为 1 时，装入程序在一致点期间将等待 I/O 操作。对于装入操作，尽管只有一个 CPU，但是如果将 CPU_PARALLELISM 设置为 2 并将 SAVECOUNT 设置为 10000，会比将 CPU_PARALLELISM 设置为 1 时完成得更快。

DISK_PARALLELISM *n*

指定 LOAD 实用程序为了将数据写入表空间容器而创建的进程数或线程数。如果未指定值，那么该实用程序将根据表空间容器数和表的特征来选择智能缺省值。

FETCH_PARALLELISM YES | NO

当执行从游标装入（该游标是使用 DATABASE 关键字声明的），或者当使用 API `sqlu_remotefetch_entry` 介质条目，并且此选项设置为 YES 时，LOAD 实用程序将尝试从远程数据源并行访存（如果可能的话）。如果设置为 NO，那么不会执行并行访存。缺省值为 YES。有关更多信息，请参阅*使用 CURSOR 文件类型来移动数据*。

INDEXING MODE

指定 LOAD 实用程序是要重建索引还是以增量方式扩展索引。有效值为:

AUTOSELECT

LOAD 实用程序将自动决定是使用 REBUILD 还是使用 INCREMENTAL 方式。应根据装入的数据量和索引树的深度来作出决定。与索引树深度相关的信息存储在索引对象中。不需要执行 RUNSTATS 来填充此信息。AUTOSELECT 是建立索引的缺省方式。

REBUILD

将重建所有索引。实用程序必须具有足够的资源来对旧的表数据和追加的表数据的所有索引键部分进行排序。

INCREMENTAL

将为索引扩充新数据。此方法将消耗索引可用空间。它只需要足够的排序空间来为已插入的记录追加索引键。只有在下列情况下才支持此方法: 当装入操作开始时，索引对象有效并且可访问（例如，在执行指定了 DEFERRED 方式的装入操作后，它将立即变得无效）。如果指定了此方式，但是因索引状态而不受支持，那么会返回警告，并且装入操作将继续采用 REBUILD 方式。同样，如果在装入构建阶段开始执行“重新启动装入”操作，那么不支持 INCREMENTAL 方式。

当满足下列所有条件时，就不支持以 INCREMENTAL 方式建立索引:

- 指定了 LOAD COPY 选项（指定了 `USEREXIT` 或 `LOGRETAIN` 选项的 *logarchmeth1*）。
- 表位于 DMS 表空间中。
- 索引对象位于一个与属于正在装入的表的其他表对象共享的表空间中。

要绕过此限制，建议您将索引放在一个单独的表空间中。

DEFERRED

如果指定此方式，那么 LOAD 实用程序将不会尝试创建索引。索引将被标记为需要刷新。首次访问与装入操作不相关的这样的索引可能会强制执行重建，或者在重新启动数据库时可能会重建索引。这种方法要求为最大索引的所有键部分提供足够的排序空间。后续用于构造索引的总时间比在 REBUILD 方式下所需要的时间更长。因此，在以 DEFERRED 方式建立索引的情况下执行多个装入操作时，（从性能方面考虑）建议您在最后一个装入操作中执行索引重建，而不允许在第一次进行非装入访问时重建索引。

仅支持对具有非唯一索引的表以 DEFERRED 方式建立索引，以便在完成装入操作后不会持久保留在装入期间插入的重复键。

ALLOW NO ACCESS

LOAD 实用程序将锁定目标表，以便在装入期间进行互斥访问。在装入期间，表的状态将设置为“正在装入”。ALLOW NO ACCESS 是缺省行为。对于 LOAD REPLACE，它是唯一有效的选项。

当表存在约束时，表的状态将设置为“设置完整性暂挂”以及“正在装入”。必须使用 SET INTEGRITY 语句来使表脱离“设置完整性暂挂”状态。

ALLOW READ ACCESS

LOAD 实用程序将锁定采用共享方式的目标表。表的状态将设置为“正在装入”和“读访问”。在装入表时，阅读器可以访问数据的非增量部分。换句话说，表的阅读器将能够访问在开始装入前就已存在的数据，而在完成装入前，正在装入的数据将不可用。ALLOW READ ACCESS 装入的 LOAD TERMINATE 或 LOAD RESTART 可以使用此选项；而 ALLOW NO ACCESS 装入的 LOAD TERMINATE 或 LOAD RESTART 不能使用此选项。而且，如果需要重建目标表的索引，那么此选项无效。

当表存在约束时，表的状态将设置为“设置完整性暂挂”以及“正在装入”和“读访问”。装入结束时，“正在装入”这种表状态将不再存在，但是“设置完整性暂挂”和“读访问”这两种表状态将继续存在。必须使用 SET INTEGRITY 语句来使表脱离“设置完整性暂挂”状态。当表处于“设置完整性暂挂”和“读访问”状态时，阅读器仍然可以访问数据的非增量部分，但是在完成 SET INTEGRITY 语句前，将仍然不可访问数据的新增（增量）部分。用户可以对同一个表执行多次装入而无须发出 SET INTEGRITY 语句。但是，在发出 SET INTEGRITY 语句前，只有原始（已检查的）数据仍然可视。

ALLOW READ ACCESS 还支持下列修饰符:

USE *tablespace-name*

如果正在重建索引，那么在 *tablespace-name* 表空间中构建了索引的影子副本，并且在 INDEX COPY PHASE 期间在装入结束时复制到了原始表空间中。只能对系统临时表空间使用此选项。如果未指定此选项，那么将在索引对象所在的同一表空间中创建影子索引。如果在索引对象所在的同一表空间中创建了影子副本，那么会立即复制影子索引对象来覆盖旧的索引对象。如果影子副本与索引对象不在同一个表空间中，那么将执行物理复制。这可能要执行大量的 I/O 操作和花费大量时间。在 INDEX COPY PHASE 期间，当装入结束时，如果表处于脱机状态，那么会进行复制。

在未使用此选项的情况下，将在原始索引所在的表空间中构建影子索引。由于原始索引和影子索引在缺省情况下同时位于同一个表空间中，因此，可能没有足够的空间用来将这两种索引保存在同一个表空间中。但是，如果使用此选项，就可以确保为索引保留足够的表空间。

如果用户不指定 INDEXING MODE REBUILD 或 INDEXING MODE AUTOSELECT，那么将忽略此选项。如果选择了 INDEXING MODE AUTOSELECT，并且 LOAD 实用程序选择以增量方式更新索引，那么也将忽略此选项。

SET INTEGRITY PENDING CASCADE

如果 LOAD 实用程序将表置于“设置完整性暂挂”状态，那么 SET INTEGRITY PENDING CASCADE 选项允许用户指定是否将已装入的表的“设置完整性暂挂”状态立即级联至所有后代（包括派生外键表、派生立即具体化查询表和派生立即登台表）。

IMMEDIATE

指示“设置完整性暂挂”状态将立即扩展至所有派生外键表、派生立即具体化查询表和派生立即登台表。对于 LOAD INSERT 操作，即使指定了 IMMEDIATE 选项，也不会将“设置完整性暂挂”状态扩展至派生外键表。

稍后，当（使用 SET INTEGRITY 语句的 IMMEDIATE CHECKED 选项）检查已装入的表是否存在约束违例时，先前处于“设置完整性暂挂读访问”状态的派生外键表将被置于“设置完整性暂挂无访问”状态。

DEFERRED

指示只有已装入的表才将处于“设置完整性暂挂”状态。而派生外键表、派生立即具体化查询表和派生立即登台表的状态将保持不变。

稍后，当（使用 SET INTEGRITY 语句的 IMMEDIATE CHECKED 选项）来检查派生外键表的父表是否发生了约束违例时，这些派生外键表可能会隐式地处于“设置完整性暂挂”状态。当检查派生立即具体化查询表和派生立即登台表的其中一个基础表是否发生了完整性违例时，这些父表可能会隐式地处于“设置完整性暂挂”状态。将发出警告（SQLSTATE 01586），指出从属表已经处于“设置完整性暂挂”状态。请参阅 SQL Reference 中 SET INTEGRITY 语句的“注释”部分，以了解这些派生表何时将处于“设置完整性暂挂”状态。

如果未指定 SET INTEGRITY PENDING CASCADE 选项:

- 则只有已装入的表才将处于“设置完整性暂挂”状态。派生外键表、派生立即具体化查询表和派生立即登台表的状态将保持不变，稍后，当检查已装入的表是否存在约束违例时，上述表可能会隐式地处于“设置完整性暂挂”状态。

如果 LOAD 实用程序并未将目标表置于“设置完整性暂挂”状态，那么说明忽略了 SET INTEGRITY PENDING CASCADE 选项。

LOCK WITH FORCE

在装入过程中，实用程序将发生各种锁定（包括表锁定）。当产生锁定时，将不采用等待策略，等待有可能会超时；此选项将允许 LOAD 实用程序强制关闭对目标表拥有相冲突锁定的其他应用程序。LOAD 实用程序不会强制关闭对系统目录表拥有相冲突锁定的应用程序。将回滚已强制关闭的应用程序，并释放

LOAD 实用程序需要的锁定。LOAD 实用程序随后就可以继续执行了。此选项与 FORCE APPLICATIONS 命令需要相同的权限（SYSADM 或 SYSCTRL）。

在装入操作开始时，ALLOW NO ACCESS 装入可能会强制挂起相冲突的锁定的应用程序。在开始装入时，LOAD 实用程序可能会强制要尝试查询或修改表的应用程序。

在装入操作开始或结束时，ALLOW READ ACCESS 装入可能会强制挂起相冲突的锁定的应用程序。在开始装入时，LOAD 实用程序可能会强制要尝试修改表的应用程序。在装入操作结束时，LOAD 实用程序可能会强制要尝试查询或修改表的应用程序。

SOURCEUSEREXIT*executable*

指定一个可执行文件的文件名，将调用该文件来为实用程序输入数据。

REDIRECT

INPUT FROM

BUFFER *input-buffer*

在 *input-buffer* 中指定的字节流被传递给用于执行给定可执行文件的进程的 STDIN 文件描述符中。

FILE *input-file*

此客户端文件的内容被传递给用于执行给定可执行文件的进程的 STDIN 文件描述符中。

OUTPUT TO

FILE *output-file*

STDOUT 和 STDERR 文件描述符被捕获至指定的标准服务器端文件。

PARALLELIZE

通过同时调用多个用户出口进程来提高进入 LOAD 实用程序的数据吞吐量。此选项仅适用于多分区数据库环境，在单一分区数据库环境中将被忽略。

有关更多信息，请参阅*使用定制应用程序（用户出口）移动数据*。

PARTITIONED DB CONFIG *partitioned-db-option*

允许您装入到一个分布在多个数据库分区中的表。PARTITIONED DB CONFIG 参数允许您指定特定于分区数据库的配置选项。*partitioned-db-option* 值可以是下列任何一项:

```
PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

在*为分区数据库环境装入配置选项*中提供了对这些选项的详细描述。

RESTARTCOUNT
保留参数。

USING *directory*
保留参数。

从 XML 文档装入数据的示例

从 XML 文档装入数据

示例 1

用户构造了数据文件，该文件包含了用于描述那些要插入表中的文档的 XDS 字段。它看起来可能类似如下:

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

对于第一行，XML 文档由文件 file1.xml 标识。注意，由于字符定界符是双引号字符，而 XDS 中存在双引号，因此 XDS 中包含的双引号是两个。对于第二行，XML 文档由文件 file2.xml 标识，从字节偏移量为 23 的位置开始，长度为 45 字节。

示例 2

用户对 XML 列发出了不带任何解析或验证选项的装入命令，并且成功地装入了数据:

```
LOAD FROM data.del of DEL INSERT INTO mytable
```

从游标装入 XML 数据

从游标装入数据与使用规则关系列类型一样。用户具有两个表（T1 和 T2），每个表都包含单个称为 C1 的 XML 列。要从 T1 装入到 T2，用户应先声明游标:

```
DECLARE X1 CURSOR FOR SELECT C1 FROM T1;
```

接着，用户可发出使用游标类型的 LOAD:

```
LOAD FROM X1 of CURSOR INSERT INTO T2
```

将特定于 XML 的 LOAD 选项装入到该游标类型与从文件装入一样。

使用说明

- 数据的装入顺序与在输入文件中的出现顺序相同。如果需要使用特定顺序，那么在尝试装入前应该对数据进行排序。如果不需要保留数据源顺序，请考虑使用 ANYORDER 文件类型修饰符（在下面的 *LOAD 实用程序的文件类型修饰符*部分作了描述）。
- LOAD 实用程序将根据现有定义来构建索引。异常表用来处理重复的唯一键。该实用程序不会强制引用完整性、执行约束检查或者更新依赖于正在装入的表的具体化查询表。包含参考约束或检查约束的表将处于“设置完整性暂挂”状态。使用 REFRESH IMMEDIATE 定义的、并且依赖于正在装入的表的总结表也将处于“设置完整性暂挂”状态。发出 SET INTEGRITY 语句以使这些表脱离“设置完整性暂挂”状态。不能对复制的具体化查询表执行装入操作。

- 如果表上存在集群索引，那么应该在执行装入前按集群索引对数据进行排序。但是，在将数据装入到多维集群（MDC）表前，不需要对数据进行排序。
- 如果在装入到受保护的表时指定了异常表，那么会将受到无效安全标号保护的任何行发送至该表。这可能会允许对异常表具有访问权的用户访问他们通常无权访问的数据。为了提高安全性，您在将对异常表的访问权授予用户时应慎重；在修复了每一行并将它复制到要装入的表中后就立即删除该行；并且在使用异常表完毕后就立即将它删除。
- 采用内部格式的安全标号可能包含换行符。如果装入使用 DEL 文件格式的文件，那么这些换行符可能会被误认为定界符。如果发生此问题，请通过在 LOAD 命令中指定文件类型修饰符 delprioritychar 来对定界符使用较旧的缺省优先级。
- 要执行使用 CURSOR 文件类型的装入操作（并且 DATABASE 关键字是在执行 DECLARE CURSOR 命令期间指定的），用来对当前连接至的数据库（用于装入）进行认证的用户标识和密码，将用于对源数据库（由 DECLARE CURSOR 命令的 DATABASE 选项指定）进行认证。如果没有指定用于连接至正在装入的数据库的用户标识或密码，那么在执行 DECLARE CURSOR 命令期间必须指定源数据库的用户标识和密码。
- 支持装入多部件 PC/IXF 文件，该文件的各个部件是从 Windows 系统复制到 AIX 系统的。所有文件的名称都必须在 LOAD 命令中指定。例如，LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1。不支持从逻辑上分割的 PC/IXF 文件装入到 Windows 操作系统。
- 重新启动失败的 LOAD 时，该行为将遵循现有行为，在现有行为中，会强制 BUILD 阶段将 REBUILD 方式用于索引。

LOAD TERMINATE 和 LOAD RESTART 字典管理的摘要

下图总结了 TERMINATE 伪指令下 LOAD 处理的压缩字典管理行为。

表 39. LOAD TERMINATE 字典管理

表 COMPRESS 属性	装入前存在字典吗?	TERMINATE: LOAD REPLACE KEEPDICTIONARY 或 LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	保留现有字典。	不保留。
YES	NO	不保留。	不保留。
NO	YES	保留现有字典。	不保留。
NO	NO	不执行任何操作。	不执行任何操作。

LOAD RESTART 将截断表直至达到最后一个一致点。如果获取最后一个 LOAD 一致点时表中存在压缩字典，那么在执行 LOAD RESTART 处理过程中，表中将存在压缩字典。在那种情况下，LOAD RESTART 将不会创建新字典。请参阅下面的表 4 以获取可能情况的摘要。

表 40. LOAD RESTART 字典管理

表 COMPRESS 属性	装入一致点前是否存在字典?	RESTART: LOAD REPLACE KEEPDICTIONARY 或 LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	保留现有字典。	保留现有字典。

表 40. LOAD RESTART 字典管理 (续)

表 COMPRESS 属性	装入一致点前是否存在字典?	RESTART: LOAD REPLACE KEEPDICTIONARY 或 LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	NO	根据 ADC 构建字典。	构建字典。
NO	YES	保留现有字典。	除去现有字典。
NO	NO	不执行任何操作。	不执行任何操作。

LOAD 实用程序的文件类型修饰符

表 41. LOAD 实用程序的有效文件类型修饰符: 所有文件格式

修饰符	描述
anyorder	此修饰符应与 *cpu_parallelism* 参数一起使用。指定不需要保留源数据顺序，这将能极大地提高 SMP 系统的性能。如果 *cpu_parallelism* 的值为 1，那么将忽略此选项。如果 SAVECOUNT > 0，那么此选项不受支持。这是因为在一致点后进行崩溃恢复将要求按顺序装入数据。
generatedignore	此修饰符通知 LOAD 实用程序: 所有生成列的数据在数据文件中都存在，但是应该忽略这些数据。这将导致所有生成列值都由该实用程序生成。此修饰符不能与 generatedmissing 或 generatedoverride 修饰符一起使用。
generatedmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含生成列的任何数据（甚至不包含 NULL）。这将导致所有生成列值都由该实用程序生成。此修饰符不能与 generatedignore 或 generatedoverride 修饰符一起使用。

generatedoverride	此修饰符指示 LOAD 实用程序接受用户为表中的所有生成列提供的数据（这与这些类型的列的正常规则相反）。从另一数据库系统中迁移数据，或者在将使用 ROLLFORWARD DATABASE 命令 RECOVER DROPPED TABLE 选项恢复的数据装入到表中时，此修饰符就很有用。当使用此修饰符时，将拒绝存在下列情况的任何行: 对于不可空的生成列，不包含任何数据或者只包含 NULL 数据（SQL3116W）。当使用此修饰符时，表将处于“设置完整性暂挂”状态。要使该表脱离“设置完整性暂挂”状态，而不验证用户提供的值，请在完成装入操作后发出以下命令:
	`SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED`
	要使该表脱离“设置完整性暂挂”状态并强制验证用户提供的值，请在完成装入操作后发出以下命令:
	`SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.`
	当指定了此修饰符，并且任何分区键、维键或分布键中有生成列时，LOAD 命令会自动将该修饰符转换为 generatedignore 并继续执行装入。这与重新生成所有生成列值具有相同的效果。
	此修饰符不能与 generatedmissing 或 generatedignore 修饰符一起使用。

identityignore	此修饰符通知 LOAD 实用程序: 标识列的数据在数据文件中已存在，但是应该忽略该数据。这将导致所有标识值都由该实用程序生成。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 identitymissing 或 identityoverride 修饰符一起使用。

表 41. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
identitymissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含标识列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 标识列与 GENERATED BY DEFAULT 标识列的行为相同。此修饰符不能与 identityignore 或 identityoverride 修饰符一起使用。
identityoverride	仅当要装入的表中存在定义为 GENERATED ALWAYS 的标识列时，才应使用此修饰符。此修饰符指示该实用程序接受这样一列的显式、非 NULL 数据（这与这些类型的标识列的正常规则相反）。从另一数据库系统中迁移数据，并且必须将表定义为 GENERATED ALWAYS 时，或者在将使用 ROLLFORWARD DATABASE 命令 DROPPED TABLE RECOVERY 选项恢复的数据装入到表中时，此修饰符很有用。当使用此修饰符时，将拒绝存在下列情况的任何行: 对于标识列，不包含任何数据或者只包含 NULL 数据（SQL3116W）。此修饰符不能与 identitymissing 或 identityignore 修饰符一起使用。当使用此选项时，LOAD 实用程序将不会尝试维护或验证表的标识列中值的唯一性。
indexfreespace=*x*	*x* 是一个 0 到 99（包括 0 和 99 在内）的整数。该值被解释为重建索引时每一索引页中要保留为可用空间的百分比。使用 INDEXING MODE INCREMENTAL 装入时将忽略此选项。可添加页面中的第一个条目，而不受限制；将添加后续条目以维持可用空间百分比阈值。缺省值是执行 CREATE INDEX 时所使用的值。 此值优先于在 CREATE INDEX 语句中指定的 PCTFREE 值；indexfreespace 选项只影响索引叶子页。
lobsinfile	*lob-path* 指定包含 LOB 数据的文件所在的路径。ASC、DEL 或 IXF 装入输入文件中包含下列文件的名称: 具有 LOB 列中的 LOB 数据的文件。 不支持将此选项与 CURSOR 文件类型一起使用。 使用“lobsinfile”修饰符时，LOBS FROM 子句指定 LOB 文件所在的位置。LOBS FROM 子句将隐式激活 LOBSINFILE 行为。LOBS FROM 子句将装入数据时要从其中搜索 LOB 文件的路径列表传递给 LOAD 实用程序。 每个路径都至少包含这样一个文件: 该文件至少包含数据文件中的“Lob 位置说明符”（LLS）所指向的一个 LOB。对于存储在 LOB 文件路径中的文件，LLS 就是对这些文件中的 LOB 所在位置的字符串表示。LLS 的格式为 *filename.ext.nnn.mmm/*，其中 *filename.ext* 是包含 LOB 的文件名，*nnn* 是文件中的 LOB 的偏移量（以字节计），*mmm* 是 LOB 的长度（以字节计）。例如，如果 db2exp.001.123.456/ 字符串存储在数据文件中，那么 LOB 位于 db2exp.001 文件中偏移量为 123 的位置，其长度为 456 字节。 要指示一个空 LOB，应将大小输入为 -1。如果将大小指定为 0，那么会将它视作长度为 0 的 LOB。对于长度为 -1 的 LOBS，将忽略偏移量和文件名。例如，空 LOB 的 LLS 可能是 db2exp.001.7.-1/。
noheader	跳过头验证代码（只适用于要装入到单一分区数据库分区组中的表中的装入操作）。 如果对位于单一分区数据库分区组中的表使用了缺省 MPP 装入（PARTITION_AND_LOAD 方式），那么不期望文件具有头。因此，不需要 noheader 修饰符。如果使用了 LOAD_ONLY 方式，那么期望文件具有头。仅当您想使用一个没有头的文件来执行 LOAD_ONLY 操作时，才需要使用 noheader 修饰符。
norowwarnings	抑制关于被拒绝行的所有警告。

表 41. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述
pagefreespace=*x*	*x* 是一个 0 到 100（包括 0 和 100 在内）的整数。该值被解释为每一数据页中要保留为可用空间的百分比。如果指定的值因最小行大小（例如，至少为 3000 字节长并且 *x* 值为 50 的一行）而变得无效，那么会将该行放在新的一页上。如果指定的值为 100，那么每一行都将位于新的一页上。一个表的 PCTFREE 值确定每页指定的可用空间量。如果尚未设置装入操作的 pagefreespace 值或者表的 PCTFREE 值，那么实用程序在每页上将填充尽可能多的空间。pagefreespace 设置的值将覆盖为表指定的 PCTFREE 值。
rowchangetimestampignore	此修饰符通知 LOAD 实用程序: ROW CHANGE TIMESTAMP 列的数据在数据文件中已存在，但是应该忽略该数据。这将导致该实用程序生成所有 ROW CHANGE TIMESTAMP。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。这意味着，对于 GENERATED ALWAYS 列，将不会拒绝任何行。此修饰符不能与 `rowchangetimestampmissing` 或 `rowchangetimestampoverride` 修饰符一起使用。
rowchangetimestampmissing	如果指定了此修饰符，那么该实用程序假定输入数据文件不包含 ROW CHANGE TIMESTAMP 列的任何数据（甚至不包含 NULL），因此将为每一行生成一个值。GENERATED ALWAYS 列与 GENERATED BY DEFAULT 列的行为相同。此修饰符不能与 `rowchangetimestampignore` 或 `rowchangetimestampoverride` 修饰符一起使用。
rowchangetimestampoverride	仅当要装入的表中存在定义为 GENERATED ALWAYS 的 ROW CHANGE TIMESTAMP 列时，才应使用此修饰符。此修饰符指示该实用程序接受这样一列的显式、非 NULL 数据（这与这些类型的 ROW CHANGE TIMESTAMP 列的正常规则相反）。从另一数据库系统中迁移数据，并且必须将表定义为 GENERATED ALWAYS 时，或者在将使用 ROLLFORWARD DATABASE 命令 DROPPED TABLE RECOVERY 选项恢复的数据装入到表中时，此修饰符很有用。当使用此修饰符时，将拒绝存在下列情况的任何行: 对于 ROW CHANGE TIMESTAMP 列，不包含任何数据或者只包含 NULL 数据（SQL3116W）。此修饰符不能与 `rowchangetimestampmissing` 或 `rowchangetimestampignore` 修饰符一起使用。当使用此选项时，LOAD 实用程序将不会尝试维护或验证表的 ROW CHANGE TIMESTAMP 列中值的唯一性。

seclabelchar	指示输入源文件中的安全标号采用安全标号值的字符串格式，而不是采用缺省编码数字格式。当装入每个安全标号时，LOAD 实用程序会将它们转换为内部格式。如果字符串的格式不正确，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3242W）。如果字符串并不表示作为用于保护表的安全策略的一部分的有效安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3243W）。
	
	如果指定了 seclabelname 修饰符，那么不能指定此修饰符，否则装入将失败并且会返回错误（SQLCODE SQL3525N）。
	
	如果一个表中仅包含单个 `DB2SECURITYLABEL` 列，那么数据文件看起来可能类似如下:
	
	`"CONFIDENTIAL:ALPHA:G2"`
	`"CONFIDENTIAL;SIGMA:G2"`
	`"TOP SECRET:ALPHA:G2"`
	
	要装入或导入此数据，必须使用 SECLABELCHAR 文件类型修饰符:
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1`

表 41. LOAD 实用程序的有效文件类型修饰符: 所有文件格式 (续)

修饰符	描述

seclabelname	指示输入源文件中的安全标号是由它们的名称指示的，而不是由缺省编码数字格式指示的。如果存在名称，那么 LOAD 实用程序会将它转换为适当的安全标号。如果不存在具有所指示的（用于保护表的）安全策略名称的安全标号，那么将不装入该行，并且将返回警告（SQLSTATE 01H53, SQLCODE SQL3244W）。
	
	如果指定了 seclabelchar 修饰符，那么不能指定此修饰符，否则装入将失败并且会返回错误（SQLCODE SQL3525N）。
	
	如果一个表中仅包含单个 DB2SECURITYLABEL 列，那么数据文件可能包含与以下内容相似的安全标号名称:
	
	`"LABEL1"`
	`"LABEL1"`
	`"LABEL2"`
	
	要装入或导入此数据，必须使用文件类型修饰符 SECLABELNAME:
	
	`LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1`
	
	注: 如果文件类型是 ASC，那么安全标号名称后面的任何空格都将被解释为该名称的一部分。为了避免此问题，可以使用文件类型修饰符 striptblanks 来确保除去空格。

totalfreespace=*x*	*x* 是一个大于或等于 0 的整数。该值被解释为: 表中的要追加至表末尾作为可用空间的总页数所占的百分比。例如，如果 *x* 为 20，而在装入数据后表中具有 100 个数据页，那么将另外追加 20 个空的数据页。表中的数据页总数就将变为 120。数据页总数并不是表中的索引页数的一部分。此选项不会影响索引对象。如果在指定此选项的情况下完成了两次装入，那么第二次装入将不会复用由第一次装入追加至末尾的额外空间。
usedefaults	如果已经指定了目标表列的源列，但是源列中不包含一个或多个行实例的数据，那么将装入缺省值。缺少的数据的示例如下: • 对于 DEL 文件: 为列值指定了两个相邻的列定界符（“,,”）或者两个相邻的列定界符之间还有任意数目的空格（“, ,”）。 • 对于 DEL/ASC/WSF 文件: 没有足够的列数或者对于原始规范来说不是足够长的行。对于 ASC 文件，并不将列值为 NULL 认为是显式丢失，也不会用缺省值来代替 NULL 列值。NULL 列值是由数字、日期、时间和时间戳记列的所有空格字符来表示的，或者是通过对任何类型的列使用 NULL INDICATOR 来指示该列为 NULL 这样来表示的。 在不使用此选项的情况下，如果源列中不包含行实例的数据，那么将发生下列情况之一: • 对于 DEL/ASC/WSF 文件: 如果该列可为空，那么将装入 NULL。如果该列不可空，那么实用程序将拒绝该行。

表 42. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）

修饰符	描述
codepage=*x*	*x* 是一个 ASCII 字符串。该值被解释为输入数据集中的数据的代码页。在执行装入操作期间，将字符数据（以及在字符中指定的数字数据）从此代码页转换为数据库代码页。 下列规则适用: • 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。 • 对于使用 EBCDIC 代码页指定的 DEL 数据来说，定界符可能与 shift-in 和 shift-out DBCS 字符不一致。 • nullindchar 必须指定标准 ASCII 代码集中包含的代码点 x20 与 x7F 之间（包括 x20 和 x7F 在内）的符号。这指的是 ASCII 符号和代码点。EBCDIC 数据可以使用相应的符号，尽管代码点将不同。 不支持将此选项与 CURSOR 文件类型一起使用。
dateformat="*x*"	*x* 是源文件中的数据所采用的格式[1]。有效日期元素包括: YYYY - 年份（四位数，范围是 0000 到 9999） M - 月份（一位数或两位数，范围是 1 到 12） MM - 月份（两位数，范围是 1 到 12; 与 M 元素互斥） D - 日（一位数或两位数，范围是 1 到 31） DD - 日（两位数，范围是 1 到 31; 与 D 元素互斥） DDD - 一年中的某日（三位数，范围是 001 到 366; 与其他日或月份元素互斥） 对于未指定的每个元素，将为它指定缺省值 1。以下是日期格式的一些示例: "D-M-YYYY" "MM.DD.YYYY" "YYYYDDD"
dumpfile = *x*	*x* 是要将被拒绝行写入的异常文件的标准名称（按照服务器数据库分区）。每条记录最多写入 32 KB 数据。以下是一个示例，说明如何指定转储文件: db2 load from data of del modified by dumpfile = /u/user/filename insert into table_name 将创建该文件，并归实例所有者所有。要覆盖缺省文件许可权，使用文件类型修饰符 dumpfileaccessall。 **注:** 1. 在分区数据库环境中，路径相对于正在装入的数据库分区来说应该是本地的，以便同时运行的装入操作不会尝试写入同一文件中。 2. 文件内容以异步缓存方式写入磁盘中。对于已失败或已中断的装入操作，在执行 LOAD RESTART 后，无法明确知道已落实到磁盘的记录数，也不能保证一致性。对于一次性启动并完成的装入操作，只能假定文件是完整的。 3. 如果指定的文件已存在，那么将不会重建该文件，但是将追加该文件。
dumpfileaccessall	当创建了转储文件时，为“OTHERS”授予读访问权。 此文件类型修饰符仅在下列情况下有效: 1. 将它与 dumpfile 文件类型修饰符一起使用 2. 用户对装入目标表具有 SELECT 特权 3. 它是在 UNIX 操作系统上的 DB2 服务器数据库分区上发出的 如果指定的文件已存在，那么将不会更改其许可权。

表 42. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
fastparse	使用时务必小心谨慎。减少对用户提供的列值进行语法检查的次数，以增强性能。确保表在结构上正确（该实用程序将执行足够的数据检查以防止分段违例或陷阱），但是，将不验证数据的一致性。只有您确信数据一致且正确时，才使用此选项。例如，如果用户提供的数据包含无效的时间戳记列值 :1>0-00-20-07.11.12.000000，那么在指定了 FASTPARSE 时，会将此值插入表，否则将拒绝此值。
implieddecimal	隐含的小数点所在的位置由列定义来确定；不再假定它位于值的末尾。例如，会将值 12345 作为 123.45 而不是 12345.00 装入 DECIMAL(8,2) 列。 此修饰符不能与 packeddecimal 修饰符一起使用。
timeformat="*x*"	*x* 是源文件中的时间格式[1]。有效时间元素包括: H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24） HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥） M - 分钟（一位数或两位数，范围是 0 到 59） MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥） S - 秒（一位数或两位数，范围是 0 到 59） SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥） SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥） TT - 正午指示符（AM 或 PM） 对于未指定的每个元素，将为它指定缺省值 0。以下是时间格式的一些示例: "HH:MM:SS" "HH.MM TT" "SSSSS"

表 42. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
timestampformat="*x*"	*x* 是源文件中的时间戳记格式[1]。有效时间戳记元素包括: YYYY - 年份（四位数，范围是 0000 到 9999） M - 月份（一位数或两位数，范围是 1 到 12） MM - 月份（两位数，范围是 01 到 12；与 M 和 MMM 元素互斥） MMM - 月份（由三个不区分大小写的字母组成的月份名称缩写；与 M 和 MM 元素互斥） D - 日（一位数或两位数，范围是 1 到 31） DD - 日（两位数，范围是 1 到 31；与 D 元素互斥） DDD - 一年中的某日（三位数，范围是 001 到 366；与其他日或月份元素互斥） H - 小时（一位数或两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24） HH - 小时（两位数，对于采用 12 小时制的系统，其范围是 0 到 12；而对于采用 24 小时制的系统，其范围是 0 到 24；此元素与 H 元素互斥） M - 分钟（一位数或两位数，范围是 0 到 59） MM - 分钟（两位数，范围是 0 到 59；此元素与 M 元素互斥） S - 秒（一位数或两位数，范围是 0 到 59） SS - 秒（两位数，范围是 0 到 59；此元素与 S 元素互斥） SSSSS - 一天当中从午夜算起已经过的秒数（五位数，范围是 00000 到 86399；此元素与其他时间元素互斥） UUUUUU - 微秒（六位数，范围是 000000 到 999999；与所有其他微秒元素互斥） UUUUU - 微秒（五位数，范围是 00000 到 99999，映射至范围 000000 到 999990；与所有其他微秒元素互斥） UUUU - 微秒（四位数，范围是 0000 到 9999，映射至范围 000000 到 999900；与所有其他微秒元素互斥） UUU - 微秒（三位数，范围是 000 到 999，映射至范围 000000 到 999000；与所有其他微秒元素互斥） UU - 微秒（两位数，范围是 00 到 99，映射至范围 000000 到 990000；与所有其他微秒元素互斥） U - 微秒（一位数，范围是 0 到 9，映射至范围 000000 到 900000；与所有其他微秒元素互斥） TT - 正午指示符（AM 或 PM）

表 42. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL）(续)

修饰符	描述

timestampformat="*x*"（续）	对于未指定的 YYYY、M、MM、D、DD 或 DDD 元素，将为它们指定缺省值 1。对于未指定的 MMM 元素，将为它指定缺省值“Jan”。对于所有其他未指定的元素，将为它们指定缺省值 0。以下是一个表示时间戳记格式的示例:
	`"YYYY/MM/DD HH:MM:SS.UUUUUU"`
	MMM 元素的有效值包括: 'jan'、'feb'、'mar'、'apr'、'may'、'jun'、'jul'、'aug'、'sep'、'oct'、'nov' 和 'dec'。这些值都不区分大小写。
	如果未指定 TIMESTAMPFORMAT 修饰符，那么 LOAD 实用程序将使用两种可能格式之一对时间戳记字段进行格式化:
	`YYYY-MM-DD-HH.MM.SS`
	`YYYY-MM-DD HH:MM:SS`
	LOAD 实用程序通过查看 DD 和 HH 之间的分隔符来选择格式。如果它是破折号“-”，那么 LOAD 实用程序将使用规则的破折号和点格式（YYYY-MM-DD-HH.MM.SS）。如果它是空格，那么 LOAD 实用程序需要使用冒号“:”来分隔 HH、MM 和 SS。
	在任一格式中，如果您包括微秒字段（UUUUUU），那么 load 实用程序将需要使用点“.”作为分隔符。YYYY-MM-DD-HH.MM.SS.UUUUUU 或 YYYY-MM-DD HH:MM:SS.UUUUUU 都是可接受的。
	以下示例说明如何将包含用户定义的日期和时间格式的数据装入到一个称为 schedule 的表中:
	`db2 load from delfile2 of del`
	`modified by timestampformat="yyyy.mm.dd hh:mm tt"`
	`insert into schedule`

usegraphiccodepage	如果给定了 `usegraphiccodepage`，那么假定装入到图形或双字节字符大对象（DBCLOB）数据字段的数据采用的是图形代码页。而假定其他数据采用的是字符代码页。图形代码页与字符代码页是相关联的。如果指定了 `codepage` 修饰符，那么 LOAD 实用程序通过该修饰符来确定字符代码页；如果未指定该修饰符，那么 LOAD 实用程序通过数据库的代码页来确定字符代码页。 仅当要恢复的表具有图形数据时，才应将此修饰符与由删除表恢复生成的定界数据文件一起使用。 **限制** 不能对 EXPORT 实用程序创建的 DEL 文件指定 `usegraphiccodepage` 修饰符，这是因为这些文件中包含只使用一种代码页编码的数据。文件中的双字节字符大对象（DBCLOB）也将忽略 `usegraphiccodepage` 修饰符。
xmlchar	指定采用字符代码页来 XML 文档进行编码。 处理采用指定的字符代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与字符代码页相匹配，否则将拒绝包含该文档的行。注意，字符代码页就是由文件类型修饰符 `codepage` 指定的值；如果未指定该修饰符，那么字符代码页就是应用程序代码页。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。

表 42. LOAD 实用程序的有效文件类型修饰符: ASCII 文件格式（ASC/DEL） (续)

修饰符	描述
xmlgraphic	指定采用指定的图形代码页来对 XML 文档进行编码。 处理采用特定的图形代码页编码、但是不包含编码声明的 XML 文档时，此选项很有用。 对于每个文档，如果存在声明标记并且包含编码属性，那么编码方式必须与图形代码页相匹配，否则将拒绝包含该文档的行。注意，图形代码页就是由文件类型修饰符 codepage 指定的值的图形组件；如果未指定该修饰符，那么图形代码页就是应用程序代码页的图形组件。缺省情况下，文档是采用 Unicode 编码的，或者它们包含具有编码属性的声明标记。

表 43. LOAD 实用程序的有效文件类型修饰符: ASC 文件格式（非定界 ASCII）

修饰符	描述
binarynumerics	数字（但不是 DECIMAL）数据必须采用二进制格式，而不能采用字符表示。这样就避免了执行成本很高的转换。 只有使用由 reclen 选项指定的定长记录的位置 ASC 才支持此选项。 下列规则适用: • 除了 BIGINT、INTEGER 和 SMALLINT 之外，其他数据类型之间都不执行转换。 • 数据长度必须与它们的目标列定义相匹配。 • FLOAT 数据必须采用 IEEE 浮点格式。 • 无论装入操作在哪个平台上运行，都认为装入源文件中的二进制数据采用大尾数法。 在此修饰符影响的列数据中不能存在 NULL。当使用此修饰符时，空白将被解释为二进制值（但通常将解释为 NULL）。
nochecklengths	如果指定了 nochecklengths，那么即使源数据的列定义超过了目标表列大小，也会尝试装入每一行。如果代码页转换导致源数据缩小，那么可以成功地装入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。
nullindchar=*x*	*x* 是单个字符。将表示空值的字符更改为 *x*。*x* 的缺省值为 Y[2]。 对于 EBCDIC 数据文件，除非字符是英文字母，否则此修饰符将区分大小写。例如，如果将空指示符指定为字母 N，那么 n 也会被识别为空指示符。

表 43. LOAD 实用程序的有效文件类型修饰符: ASC 文件格式（非定界 ASCII）(续)

修饰符	描述

packeddecimal	直接装入压缩十进制数据，这是因为 binarynumerics 修饰符不包括 DECIMAL 字段类型。
	
	只有使用由 reclen 选项指定的定长记录的位置 ASC 才支持此选项。
	
	带符号的半字节的支持值包括:
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	在此修饰符影响的列数据中不能存在 NULL。当使用此修饰符时，空白将被解释为二进制值（但通常将解释为 NULL）。
	
	无论在哪个服务器平台上，都认为装入源文件中的二进制数据的字节顺序是大尾数法；也就是说，当在 Windows 操作系统上使用此修饰符时，不能保留字节顺序。
	
	此修饰符不能与 implieddecimal 修饰符一起使用。

reclen=*x*	*x* 是一个整数，最大值为 32 767。会读取每行的 *x* 个字符，但未使用换行符来指示行的末尾。
striptblanks	当将数据装入到一个变长字段时，将截断任何尾部空格。如果未指定此选项，那么将保留空格。 不能将此选项与 striptnulls 同时指定。它们是互斥选项。此选项将替换过时的 t 选项，支持此过时选项只是为了保持与早前版本的兼容性。
striptnulls	当将数据装入到一个变长字段时，将截断任何尾部 NULL（0x00 字符）。如果未指定此选项，那么将保留 NULL。 不能将此选项与 striptblanks 同时指定。它们是互斥选项。此选项将替换过时的 padwithzero 选项，支持此过时选项只是为了保持与早前版本的兼容性。

zoneddecimal	装入分区十进制数据，这是因为 BINARYNUMERICS 修饰符不包括 DECIMAL 字段类型。只有使用由 RECLEN 选项指定的定长记录的位置 ASC 才支持此选项。
	
	带符号的半字节值可以是下列其中一项:
	`+ = 0xC 0xA 0xE 0xF`
	`- = 0xD 0xB`
	
	受支持的位数值是 0x0 到 0x9。
	
	受支持的区域值是 0x3 和 0xF。

表 44. LOAD 实用程序的有效文件类型修饰符: DEL 文件格式（定界 ASCII）

修饰符	描述

chardel*x*	*x* 是单个字符串定界符。缺省值是双引号（"）。使用指定的字符而不是使用双引号将字符串引起来[23]。如果您想显式地指定双引号（"）作为字符串定界符，那么应按如下所示指定双引号:
	`modified by chardel""`
	
	也可以指定单引号（'）作为字符串定界符，如下所示:
	`modified by chardel''`

coldel*x*	*x* 是一个单字符列定界符。缺省值是逗号（,）。使用指定字符而不是逗号来表示列的末尾[23]。

表 44. LOAD 实用程序的有效文件类型修饰符: DEL 文件格式（定界 ASCII）(续)

修饰符	描述
decplusblank	加号字符。导致在正的十进制值前面加上空格而不是加号（+）。缺省操作是在正的十进制值前面加上加号。
decpt*x*	*x* 是单个字符，它取代句点作为小数点字符。缺省值是句点（.）。使用指定字符而不是句点作为小数点字符[23]。

delprioritychar	当前，定界符的缺省优先级为记录定界符、字符定界符和列定界符。此修饰符通过将定界符优先级还原为字符定界符、记录定界符和列定界符，来保护依赖于旧的优先级的现有应用程序。语法:
	`db2 load ... modified by delprioritychar ...`
	例如，用以下 DEL 数据文件作为示例:
	`"Smith, Joshua",4000,34.98<row delimiter>`
	`"Vincent,<row delimiter>, is a manager", ...`
	`... 4005,44.37<row delimiter>`
	如果指定了 delprioritychar 修饰符，那么此数据文件中将只有两行。第二个 <row delimiter> 将被解释为第二行的第一个数据列，而第一个和第三个 <row delimiter> 被解释为实际的记录定界符。如果未指定此修饰符，那么此数据文件中将有三行，每一行都用 <row delimiter> 定界。

keepblanks	保留类型为 CHAR、VARCHAR、LONG VARCHAR 或 CLOB 的每个字段中的前导空格和尾部空格。如果未指定此选项，那么会除去字符定界符外部的所有前导空格和尾部空格，并且会在表中插入 NULL 表示所有空白字段。
	以下示例说明如何将数据装入到称为 TABLE1 的表中，同时还要保留数据文件中的所有前导空格和尾部空格:
	`db2 load from delfile3 of del`
	`modified by keepblanks`
	`insert into table1`

nochardel	LOAD 实用程序将假定在列定界符之间找到的所有字节都是列数据的一部分。字符定界符将被解析为列数据的一部分。如果数据是使用 DB2 导出的，那么不应指定此选项（除非在导出时指定了 nochardel）。提供此修饰符的目的是支持不具有字符定界符的供应商数据文件。未正确使用此修饰符可能会导致数据丢失或毁坏。 不能将此选项与 chardelx、delprioritychar 或 nodoubledel 同时指定。它们是互斥选项。
nodoubledel	不识别双字符定界符。

表 45. LOAD 实用程序的有效文件类型修饰符: IXF 文件格式

修饰符	描述
forcein	指示实用程序接受数据（即使代码页不匹配也接受），并且阻止代码页之间进行转换。 将检查定长目标字段，以验证它们对于数据来说是否足够大。如果指定了 nochecklengths，那么不会进行检查，并且将尝试装入每一行。
nochecklengths	如果指定了 nochecklengths，那么即使源数据的列定义超过了目标表列大小，也会尝试装入每一行。如果代码页转换导致源数据缩小，那么可以成功地装入这些行；例如，源中的 4 字节 EUC 数据在目标中可以缩小为 2 字节的 DBCS 数据，因此只需要一半的空间。如果明确知道源数据将适合于所有情况，无论列定义是否相匹配都是如此，那么此选项将特别有用。

注:

1. 日期格式字符串两边必须具有双引号。字段分隔符不能包含下列任何字符: a-z、A-Z 和 0-9。字段分隔符不应与 DEL 文件格式中的字符定界符或字段定界符相同。如果元素的开始和结束位置是明确的，那么字段分隔符是可选的。如果使用了诸如 D、H、M 或 S 之类的元素（取决于修饰符），那么由于条目长度是可变的，因此可能存在不明确性。

   对于时间戳记格式，必须要注意避免月份描述符与分钟描述符之间的不明确性，这是因为它们都使用字母 M。月份字段必须与其他日期字段相邻。而分钟字段必须与其他时间字段相邻。以下是一些不明确的时间戳记格式:

   ```
   "M"（既可能是月份，也可能是分钟）
      "M:M"（无法区分哪个是月份，哪个是分钟）
      "M:YYYY:M"（两者都将被解释为月份。）
      "S:M:YYYY"（与时间值和日期值都相邻）
   ```

   在不明确的情况下，实用程序将报告一条错误消息，并且操作将失败。

   以下是一些明确的时间戳记格式:

   ```
   "M:YYYY"（表示月份）
      "S:M"（表示分钟）
      "M:YYYY:S:M"（前者表示月份，后者表示分钟）
      "M:H:YYYY:M:D"（前者表示分钟，后者表示月份）
   ```

   在某些字符（例如，双引号和反斜杠）前面必须添加转义字符（例如，@2329。

2. 为 chardel、coldel 或 decpt 文件类型修饰符指定的字符值必须采用源数据的代码页来指定。

   可以使用 xJJ 或 0xJJ 语法来指定字符代码点（而不是字符符号）。其中 JJ 是代码点的十六进制表示法。例如，要指定 # 字符作为列定界符，可使用下列方法之一:

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

3. *移动数据时的定界符注意事项*列示了可以用作定界符的字符存在的限制。
4. 如果您尝试将不受支持的文件类型与 MODIFIED BY 选项配合使用，那么 LOAD 实用程序将不会发出警告。如果尝试这样做，那么装入操作将失败，并且会返回错误代码。
5. 导入到包含隐式隐藏的 ROW CHANGE TIMESTAMP 列的表中时，将不采用该列的隐式隐藏属性。因此，如果列的数据在要导入的数据中不存在，且不存在任何显式列列表，那么*必须*在导入命令中指定 rowchangetimestampmissing 文件类型修饰符。

表 46. 同时使用 codepage 和 usegraphiccodepage 时的 LOAD 行为

codepage=N	usegraphiccodepage	LOAD 行为
缺少	缺少	假定文件中的所有数据都采用数据库代码页，而不采用应用程序代码页，即使指定了 CLIENT 选项亦如此。
存在	缺少	假定文件中的所有数据都采用代码页 N。 **警告**: 如果 N 是单字节代码页，那么将图形数据装入到数据库中时将毁坏该数据。

表 46. 同时使用 codepage 和 usegraphiccodepage 时的 LOAD 行为 (续)

codepage=N	usegraphiccodepage	LOAD 行为
缺少	存在	假定文件中的字符数据采用数据库代码页，即使指定了 CLIENT 选项亦如此。假定图形数据要采用数据库图形数据的代码页，即使指定了 CLIENT 选项亦如此。 如果数据库代码页是单字节，那么假定所有数据都采用数据库代码页。 **警告**: 将图形数据装入到单字节数据库中时将毁坏该数据。
存在	存在	假定字符数据采用代码页 N。假定图形数据采用图形代码页 N。 如果 N 是单字节或双字节代码页，那么假定所有数据都采用代码页 N。 **警告**: 如果 N 是单字节代码页，那么将图形数据装入到数据库中时将毁坏该数据。

db2Load - 将数据装入到表中

将数据装入 DB2 表中。服务器上的数据可能以文件、游标、磁带或命名管道形式存在。位于远程连接的客户机上的数据可能以标准文件、游标或命名管道形式存在。尽管 LOAD 实用程序比 IMPORT 实用程序速度更快，但是，LOAD 实用程序不支持在层次结构级别装入数据或者将数据装入昵称中。

权限

为下列其中一项:

- *sysadm*
- *dbadm*
- 对数据库的 LOAD 权限，以及下列特权:
  - 以 INSERT 方式、TERMINATE 方式（用于终止先前的装入插入操作）或 RESTART 方式（用于重新启动先前的装入插入操作）调用 LOAD 实用程序时，对表的 INSERT 特权
  - 以 REPLACE 方式、TERMINATE 方式（用于终止先前的装入替换操作）或 RESTART 方式（用于重新启动先前的装入替换操作）调用 LOAD 实用程序时，对表的 INSERT 和 DELETE 特权
  - 作为装入操作使用异常表时，对异常表的 INSERT 特权

注: 通过，所有装入进程和所有 DB2 服务器进程都归实例所有者所有。所有这些进程都使用实例所有者的标识来访问所需要的文件。因此，无论任何人调用该命令，实例所有者都必须对输入文件具有读访问权。

必需的连接

数据库。如果启用了隐式连接，那么将建立与缺省数据库的连接。如果从 Linux、UNIX 或 Windows 客户机对 Linux、UNIX 或 Windows 数据库服务器进行实用程序访问，必须通过引擎进行直接连接，而不能通过 DB2 Connect 网关或环回环境进行连接。

实例。不需要显式连接。如果已经与数据库建立了连接，那么会尝试与本地实例进行隐式连接。

API 包含文件

```
db2ApiDf.h
```

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
    db2Load (
      db2Uint32 versionNumber,
      void * pParmStruct,
      struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadStruct
{
      struct sqlu_media_list *piSourceList;
      struct sqlu_media_list *piLobPathList;
      struct sqldcol *piDataDescriptor;
      struct sqlchar *piActionString;
      char *piFileType;
      struct sqlchar *piFileTypeMod;
      char *piLocalMsgFileName;
      char *piTempFilesPath;
      struct sqlu_media_list *piVendorSortWorkPaths;
      struct sqlu_media_list *piCopyTargetList;
      db2int32 *piNullIndicators;
      struct db2LoadIn *piLoadInfoIn;
      struct db2LoadOut *poLoadInfoOut;
      struct db2PartLoadIn *piPartLoadInfoIn;
      struct db2PartLoadOut *poPartLoadInfoOut;
      db2int16 iCallerAction;
      struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2LoadStruct;

typedef SQL_STRUCTURE db2LoadUserExit
{
      db2Char iSourceUserExitCmd;
      struct db2Char *piInputStream;
      struct db2Char *piInputFileName;
      struct db2Char *piOutputFileName;
      db2Uint16 *piEnableParallelism;
} db2LoadUserExit;

typedef SQL_STRUCTURE db2LoadIn
{
      db2Uint64 iRowcount;
      db2Uint64 iRestartcount;
      char *piUseTablespace;
      db2Uint32 iSavecount;
      db2Uint32 iDataBufferSize;
      db2Uint32 iSortBufferSize;
      db2Uint32 iWarningcount;
      db2Uint16 iHoldQuiesce;
      db2Uint16 iCpuParallelism;
      db2Uint16 iDiskParallelism;
      db2Uint16 iNonrecoverable;
```

```
      db2Uint16 iIndexingMode;
      db2Uint16 iAccessLevel;
      db2Uint16 iLockWithForce;
      db2Uint16 iCheckPending;
      char iRestartphase;
      char iStatsOpt;
      db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
      db2Uint16 iSetIntegrityPending;
      struct db2LoadUserExit *piSourceUserExit;
} db2LoadIn;

typedef SQL_STRUCTURE db2LoadOut
{
      db2Uint64 oRowsRead;
      db2Uint64 oRowsSkipped;
      db2Uint64 oRowsLoaded;
      db2Uint64 oRowsRejected;
      db2Uint64 oRowsDeleted;
      db2Uint64 oRowsCommitted;
} db2LoadOut;

typedef SQL_STRUCTURE db2PartLoadIn
{
      char *piHostname;
      char *piFileTransferCmd;
      char *piPartFileLocation;
      struct db2LoadNodeList *piOutputNodes;
      struct db2LoadNodeList *piPartitioningNodes;
      db2Uint16 *piMode;
      db2Uint16 *piMaxNumPartAgents;
      db2Uint16 *piIsolatePartErrs;
      db2Uint16 *piStatusInterval;
      struct db2LoadPortRange *piPortRange;
      db2Uint16 *piCheckTruncation;
      char *piMapFileInput;
      char *piMapFileOutput;
      db2Uint16 *piTrace;
      db2Uint16 *piNewline;
      char *piDistfile;
      db2Uint16 *piOmitHeader;
      SQL_PDB_NODE_TYPE *piRunStatDBPartNum;
} db2PartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
      SQL_PDB_NODE_TYPE *piNodeList;
      db2Uint16 iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
      db2Uint16 iPortMin;
      db2Uint16 iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
      db2Uint64 oRowsRdPartAgents;
      db2Uint64 oRowsRejPartAgents;
      db2Uint64 oRowsPartitioned;
      struct db2LoadAgentInfo *poAgentInfoList;
      db2Uint32 iMaxAgentInfoEntries;
      db2Uint32 oNumAgentInfoEntries;
} db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
```

```
{
      db2int32 oSqlcode;
      db2Uint32 oTableState;
      SQL_PDB_NODE_TYPE oNodeNum;
      db2Uint16 oAgentType;
} db2LoadAgentInfo;

SQL_API_RC SQL_API_FN
    db2gLoad (
      db2Uint32 versionNumber,
      void * pParmStruct,
      struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadStruct
{
      struct sqlu_media_list *piSourceList;
      struct sqlu_media_list *piLobPathList;
      struct sqldcol *piDataDescriptor;
      struct sqlchar *piActionString;
      char *piFileType;
      struct sqlchar *piFileTypeMod;
      char *piLocalMsgFileName;
      char *piTempFilesPath;
      struct sqlu_media_list *piVendorSortWorkPaths;
      struct sqlu_media_list *piCopyTargetList;
      db2int32 *piNullIndicators;
      struct db2gLoadIn *piLoadInfoIn;
      struct db2LoadOut *poLoadInfoOut;
      struct db2gPartLoadIn *piPartLoadInfoIn;
      struct db2PartLoadOut *poPartLoadInfoOut;
      db2int16 iCallerAction;
      db2Uint16 iFileTypeLen;
      db2Uint16 iLocalMsgFileLen;
      db2Uint16 iTempFilesPathLen;
      struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2gLoadStruct;

typedef SQL_STRUCTURE db2gLoadIn
{
      db2Uint64 iRowcount;
      db2Uint64 iRestartcount;
      char *piUseTablespace;
      db2Uint32 iSavecount;
      db2Uint32 iDataBufferSize;
      db2Uint32 iSortBufferSize;
      db2Uint32 iWarningcount;
      db2Uint16 iHoldQuiesce;
      db2Uint16 iCpuParallelism;
      db2Uint16 iDiskParallelism;
      db2Uint16 iNonrecoverable;
      db2Uint16 iIndexingMode;
      db2Uint16 iAccessLevel;
      db2Uint16 iLockWithForce;
      db2Uint16 iCheckPending;
      char iRestartphase;
      char iStatsOpt;
      db2Uint16 iUseTablespaceLen;
      db2Uint16 iSetIntegrityPending;
      db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
      struct db2LoadUserExit *piSourceUserExit;
} db2gLoadIn;

typedef SQL_STRUCTURE db2gPartLoadIn
{
      char *piHostname;
```

```
      char *piFileTransferCmd;
      char *piPartFileLocation;
      struct db2LoadNodeList *piOutputNodes;
      struct db2LoadNodeList *piPartitioningNodes;
      db2Uint16 *piMode;
      db2Uint16 *piMaxNumPartAgents;
      db2Uint16 *piIsolatePartErrs;
      db2Uint16 *piStatusInterval;
      struct db2LoadPortRange *piPortRange;
      db2Uint16 *piCheckTruncation;
      char *piMapFileInput;
      char *piMapFileOutput;
      db2Uint16 *piTrace;
      db2Uint16 *piNewline;
      char *piDistfile;
      db2Uint16 *piOmitHeader;
      void *piReserved1;
      db2Uint16 iHostnameLen;
      db2Uint16 iFileTransferLen;
      db2Uint16 iPartFileLocLen;
      db2Uint16 iMapFileInputLen;
      db2Uint16 iMapFileOutputLen;
      db2Uint16 iDistfileLen;
} db2gPartLoadIn;


/* Definitions for iUsing value of db2DMUXmlValidate structure               */
#define DB2DMU_XMLVAL_XDS              1         /* Use XDS                  */
#define DB2DMU_XMLVAL_SCHEMA           2         /* Use a specified schema   */
#define DB2DMU_XMLVAL_SCHEMALOC_HINTS  3         /* Use schemaLocation hints */
#define DB2DMU_XMLVAL_ORIGSCHEMA       4         /* Use schema that document was
                                                    originally validated against
                                                    (load from cursor only) */
```

db2Load API 参数

versionNumber

输入。指定作为第二个参数 pParmStruct 传递的结构的版本和发行版级别。

pParmStruct

输入。指向 db2LoadStruct 结构的指针。

pSqlca

输出。指向 sqlca 结构的指针。

db2LoadStruct 数据结构参数

piSourceList

输入。指向 sqlu_media_list 结构的指针，该结构用来提供源文件、设备、供应商、管道或 SQL 语句的列表。

此结构中提供的信息取决于 media_type 字段的值。有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是:

SQLU_SQL_STMT

如果 media_type 字段设置为此值，那么调用者将通过目标字段的 pStatement 字段来提供 SQL 查询。pStatement 字段的类型为 sqlu_statement_entry。由于 LOAD 实用程序在每次装入时只接受单个 SQL 查询，因此必须将会话字段的值设置为 1。

SQLU_SERVER_LOCATION

如果 media_type 字段设置为此值，那么调用者将通过 sqlu_location_entry

结构来提供信息。会话字段指示所提供的 sqlu_location_entry 结构的个数。此参数用于文件、设备和命名管道。

SQLU_CLIENT_LOCATION

如果 media_type 字段设置为此值，那么调用者将通过 sqlu_location_entry 结构来提供信息。会话字段指示所提供的 sqlu_location_entry 结构的个数。此参数用于标准文件和命名管道。注意，仅当通过远程连接的客户机调用 API 时，此 media_type 才有效。

SQLU_TSM_MEDIA

如果 media_type 字段设置为此值，那么将使用 sqlu_vendor 结构，其中 filename 是要装入的数据的唯一标识。无论会话的值是什么，都只能有一个 sqlu_vendor 条目。会话字段指示要启动的 TSM 会话的个数。LOAD 实用程序将使用不同的序号来启动会话，但是将使用一个 sqlu_vendor 条目中的相同数据。

SQLU_OTHER_MEDIA

如果 media_type 字段设置为此值，那么将使用 sqlu_vendor 结构，其中 shr_lib 是共享库名，filename 是要装入的数据的唯一标识。无论会话的值是什么，都只能有一个 sqlu_vendor 条目。会话字段指示要启动的其他供应商会话的个数。LOAD 实用程序将使用不同的序号来启动会话，但是将使用一个 sqlu_vendor 条目中的相同数据。

SQLU_REMOTEFETCH

如果 media_type 字段设置为此值，那么调用者将通过 sqlu_remotefetch_entry 结构来提供信息。必须将会话字段的值设置为 1。

piLobPathList

输入。指向 sqlu_media_list 结构的指针。对于 IXF、ASC 和 DEL 文件类型，这是标准路径或设备的列表，用来标识要装入的各个 LOB 文件所在的位置。将在 IXF、ASC 或 DEL 文件中找到文件名，并将它们追加至所提供的路径。

此结构中提供的信息取决于 media_type 字段的值。有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是:

SQLU_LOCAL_MEDIA

如果设置为此值，那么调用者将通过 sqlu_media_entry 结构来提供信息。会话字段指示所提供的 sqlu_media_entry 结构的个数。

SQLU_TSM_MEDIA

如果设置为此值，那么使用了 sqlu_vendor 结构，其中 filename 是要装入的数据的唯一标识。无论会话的值是什么，都只能有一个 sqlu_vendor 条目。会话字段指示要启动的 TSM 会话的个数。LOAD 实用程序将使用不同的序号来启动会话，但是将使用一个 sqlu_vendor 条目中的相同数据。

SQLU_OTHER_MEDIA

如果设置为此值，那么将使用 sqlu_vendor 结构，其中 shr_lib 是共享库名，filename 是要装入的数据的唯一标识。无论会话的值是什么，都只能有一个 sqlu_vendor 条目。会话字段指示要启动的其他供应商会话的个数。LOAD 实用程序将使用不同的序号来启动会话，但是将使用一个 sqlu_vendor 条目中的相同数据。

piDataDescriptor

输入。指向 sqldcol 结构的指针，该结构中包含有关所选择的要从外部文件装入的列的信息。

如果将 piFileType 参数设置为 SQL_ASC，那么必须将此结构的 dcolmeth 字段设置为 SQL_METH_L。用户为要装入的每一列指定开始和结束位置。

如果文件类型为 SQL_DEL，那么 dcolmeth 可以为 SQL_METH_P 或 SQL_METH_D。如果为 SQL_METH_P，那么用户必须提供源列位置。如果为 SQL_METH_D，那么会将文件中的第一列装入到表的第一列，以此类推。

如果文件类型为 SQL_IXF，那么 dcolmeth 可以为 SQL_METH_P、SQL_METH_D 或 SQL_METH_N。用于 DEL 文件的规则在此处也适用，只不过 SQL_METH_N 指示要在 sqldcol 结构中提供的文件列名。

piActionString

不推荐使用。替换为 piLongActionString。

piLongActionString

输入。指向 *sqllob* 结构的指针，该结构包含长度为 4 字节的字段，后跟用来指定会影响表的操作的字符数组。

字符数组的格式如下所示:

```
"INSERT|REPLACE KEEPDICTIONARY|REPLACE RESETDICTIONARY|RESTART|TERMINATE
INTO tbname [(column_list)]
[FOR EXCEPTION e_tbname]"
```

INSERT

将已装入的数据添加至表，但不更改现有表数据。

REPLACE

从表中删除现有的所有数据，然后插入已装入的数据。表定义和索引定义不会发生更改。

RESTART

重新启动先前已中断的装入操作。装入操作将自动从装入、构建或删除阶段中的最后一个一致点继续执行。

TERMINATE

终止先前已中断的装入操作，并将该操作回滚到开始执行它时的时间点，即使已经超过了一致点亦如此。该操作中涉及到的任何表空间的状态将恢复为正常状态，并且会使所有表对象保持一致（索引对象可能会被标记为无效。在这种情况下，下一次进行访问时将自动重建索引）。如果表所在的表空间不处于装入暂挂状态，那么此选项不会影响表空间的状态。

装入终止选项不会使表空间脱离“备份暂挂”状态。

tbname

要将数据装入到的表的名称。该表不能是系统表或者已声明临时表。可指定别名，或者指定标准表名或非限定表名。标准表名的格式为 schema.tablename。如果指定了非限定表名，那么将使用 CURRENT SCHEMA 语句来限定表。

(column_list)

要将数据插入到的表列名的列表。列名之间必须用逗号分隔。如果列名中包含空格或小写字符，那么必须用引号将它引起来。

FOR EXCEPTION e_tbname

指定要将错误行复制到的异常表。异常表用来存储违反唯一索引规则、范围限制和安全策略的行的副本。

NORANGEEXC

指示如果某行因违反范围而被拒绝，那么不会将该行插入到异常表中。

NOUNIQUEEXC

指示如果某行因违反唯一约束而被拒绝，那么不会将该行插入到异常表中。

piFileType

输入。一个字符串，用来指示输入数据源的格式。受支持的外部格式是（这些格式是在 sqlutil 中定义的）：

SQL_ASC

非定界 ASCII。

SQL_DEL

定界 ASCII，用于与 dBase、BASIC 和 IBM Personal Decision Series 程序以及许多其他数据库管理器和文件管理器进行交换。

SQL_IXF

PC 版本的集成交换格式，这是用于导出表中的数据，以便稍后可将该数据装入同一个表或者装入另一个数据库管理器表的首选方法。

SQL_CURSOR

SQL 查询。通过 piSourceList 参数传入的 sqlu_media_list 结构属于 SQLU_SQL_STMT 或 SQLU_REMOTEFETCH 类型，且该结构引用一个 SQL 查询或表名。

piFileTypeMod

输入。指向 sqlchar 结构的指针，该结构后跟用来指定一个或多个处理选项的一组字符。如果此指针为 NULL，或者指向的结构未包含任何字符，那么会将此操作解释为选择缺省规范。

并不是所有选项都可用于所有受支持的文件类型。请参阅相关链接 -“LOAD 实用程序的文件类型修饰符”。

piLocalMsgFileName

输入。一个字符串，它包含要将输出消息写入的本地文件的名称。

piTempFilesPath

输入。一个字符串，它指定服务器上的要用于临时文件的路径名。创建临时文件是为了存储消息、一致点和删除阶段信息。

piVendorSortWorkPaths

输入。指向 sqlu_media_list 结构的指针，该结构指定“供应商排序”工作目录。

piCopyTargetList

输入。指向 sqlu_media_list 结构的指针，（如果要创建副本映像）该结构用来提供要将副本映像写入的目标路径、设备或共享库的列表。

此结构中提供的值取决于 media_type 字段的值。此参数的有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是:

SQLU_LOCAL_MEDIA

如果要将副本写入本地介质，那么将 media_type 设置为此值，并提供有关 sqlu_media_entry 结构中的目标的信息。会话字段指定所提供的 sqlu_media_entry 结构的个数。

SQLU_TSM_MEDIA

如果要将副本写入 TSM，那么使用此值。不需要其他信息。

SQLU_OTHER_MEDIA

如果要使用供应商产品，那么使用此值，并通过 sqlu_vendor 结构来提供更多信息。将此结构的 shr_lib 字段设置为供应商产品的共享库名称。无论会话的值是什么，都只提供一个 sqlu_vendor 条目。会话字段指定所提供的 sqlu_media_entry 结构的个数。LOAD 实用程序将使用不同的序号来启动会话，但是将使用一个 sqlu_vendor 条目中提供的相同数据。

piNullIndicators

输入。仅适用于 ASC 文件。用来指示列数据是否可空的整数数组。此数组中的元素与从数据文件中装入的列之间存在一一对应关系。也就是说，元素的个数必须与 piDataDescriptor 参数的 dcolnum 字段的值相等。该数组中的每个元素都包含一个数字，用来标识数据文件中要用作空指示符字段的位置。该数字也可以是零，指示表列是不可空的。如果元素不为零，那么数据文件中所标识的位置必须包含 Y 或 N。如果为 Y，那么表示表列数据是 NULL；如果为 N，那么表示表列数据不是 NULL。

piLoadInfoIn

输入。指向 db2LoadIn 结构的指针。

poLoadInfoOut

输出。指向 db2LoadOut 结构的指针。

piPartLoadInfoIn

输入。指向 db2PartLoadIn 结构的指针。

poPartLoadInfoOut

输出。指向 db2PartLoadOut 结构的指针。

iCallerAction

输入。调用者请求的操作。有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是:

SQLU_INITIAL

初始调用。在首次调用 API 时必须使用此值（或 SQLU_NOINTERRUPT）。

SQLU_NOINTERRUPT

初始调用。不暂挂处理。在首次调用 API 时必须使用此值（或 SQLU_INITIAL）。

如果初始调用或任何后续调用已返回，但是要求进行调用的应用程序执行某些操作后才能完成所请求的装入操作，那么必须将调用者操作设置为下列其中一项:

SQLU_CONTINUE

继续处理。在初始调用返回的结果是实用程序要求用户输入（例如，

要求对磁带条件结束作出响应）后，只能在对 API 的后续调用中使用此值。它指定实用程序所请求的用户操作已完成，该实用程序可以继续处理初始请求。

SQLU_TERMINATE
终止处理。导致 LOAD 实用程序过早退出，并使正在装入的表空间处于 LOAD_PENDING 状态。如果不需要对数据进行进一步处理，那么应指定此选项。

SQLU_ABORT
终止处理。导致 LOAD 实用程序过早退出，并使正在装入的表空间处于 LOAD_PENDING 状态。如果不需要对数据进行进一步处理，那么应指定此选项。

SQLU_RESTART
重新开始处理。

SQLU_DEVICE_TERMINATE
终止单个设备。如果该实用程序要停止从设备中读取数据，但是要对数据进行进一步处理，那么应指定此选项。

piXmlPathList
输入。指向 sqlu_media_list 的指针，它的 media_type 字段设置为 SQLU_LOCAL_MEDIA，并且它的 sqlu_media_entry 结构列示客户机上用于存储 xml 文件的路径。

db2LoadUserExit 数据结构参数

iSourceUserExitCmd
输入。将用来为实用程序提供数据的可执行文件的标准名称。为了安全起见，必须将可执行文件放在服务器上的 sqllib/bin 目录中。如果 piSourceUserExit 不为 NULL，那么必须使用此参数。

piInputStream、piInputFileName、piOutputFileName 和 piEnableParallelism 字段是可选的。

piInputStream
输入。这是一个类属字节流，将通过 STDIN 直接传递给用户出口应用程序。您完全可以控制此字节流中包含的数据以及数据的格式。LOAD 实用程序只是将此字节流转移到服务器，并通过填充进程的 STDIN 来将字节流传递给用户出口应用程序（LOAD 实用程序将不会转换代码页或者修改字节流）。用户出口应用程序将从 STDIN 中读取自变量，但是将使用期望的数据。

此功能部件的一个重要属性是能够隐藏敏感信息（例如，用户标识/密码）。

piInputFileName
输入。包含标准客户端文件的名称，将通过填充进程的 STDIN 来将该文件的内容传递给用户出口应用程序。

piOutputFileName
输入。服务器端文件的标准名称。正在执行用户出口应用程序的进程的 STDOUT 流和 STDERR 流都将保存在此文件中。当 piEnableParallelism 为 TRUE 时，将创建多个文件（为每个用户出口实例创建一个文件），并且将对每个文件名追加一个 3 位数的节点号值（例如，<filename>.000）。

piEnableParallelism

输入。一个标志，它指示实用程序应尝试并行调用用户出口应用程序。

db2LoadIn 数据结构参数

iRowcount

输入。要装入的物理记录数。允许用户只装入文件中前面的 rowcnt 行。

iRestartcount

输入。保留以备将来使用。

piUseTablespace

输入。如果正在重建索引，那么在 iUseTablespaceName 表空间中构建了索引的影子副本，并且在装入结束时复制到原始表空间中。只能对系统临时表空间使用此选项。如果未指定此选项，那么将在索引对象所在的同一表空间中创建影子索引。

如果在索引对象所在的同一表空间中创建了影子副本，那么会立即复制影子索引对象来覆盖旧的索引对象。如果影子副本与索引对象不在同一个表空间中，那么将执行物理复制。这可能要执行大量的 I/O 操作和花费大量时间。当装入结束时，如果表处于脱机状态，那么会进行复制。

如果 iAccessLevel 为 SQLU_ALLOW_NO_ACCESS，那么会忽略此字段。

如果用户不指定 INDEXING MODE REBUILD 或 INDEXING MODE AUTOSELECT，那么将忽略此选项。如果选择了 INDEXING MODE AUTOSELECT，并且 LOAD 实用程序选择以增量方式更新索引，那么也将忽略此选项。

iSavecount

要在建立一致点前装入的记录数。此值被转换为页面计数，并且向上取整为扩展数据块大小的间隔。由于在每个一致点都会发出消息，因此，如果将使用“db2LoadQuery - 装入查询”来监视装入操作，那么应选择此选项。如果 savecount 的值不是足够大，那么在每个一致点执行的活动的同步将影响性能。

缺省值为 0，这意味着除非确实需要建立一致点，否则将不会建立任何一致点。

iDataBufferSize

要用作缓存空间的 4KB 页数（不考虑并行度），将使用该缓存空间传送实用程序中的数据。如果指定值小于算术最小值，那么将使用需要的最小值，但不会返回警告。

此内存是直接从实用程序堆中分配的，可通过数据库配置参数 util_heap_sz 来修改此内存大小。

如果未指定值，那么实用程序在运行时将计算智能缺省值。该缺省值取决于在实例化装入程序时实用程序堆中的可用空间以及表的某些特征。

iSortBufferSize

输入。此选项指定一个在装入操作期间将覆盖 SORTHEAP 数据库配置参数的值。仅当装入具有索引的表以及 iIndexingMode 参数未指定为 SQLU_INX_DEFERRED 时，此选项才有效。为此选项指定的值不能超过 SORTHEAP 的值。此参数对于在不更改 SORTHEAP 值的情况下调整 LOAD 实用程序所使用的排序内存是很有用的，这还会影响常规查询处理。

iWarningcount

输入。在发出 warningcnt 个警告后就停止装入操作。如果希望不产生警告，那

么设置此参数，但是需要验证使用的是正确的文件和表。如果未正确指定装入文件或目标表，那么 LOAD 实用程序将对它试图装入的每一行生成一个警告，这将导致装入失败。如果 warningcnt 为 0，或者未指定此选项，那么无论发出多少个警告，都将继续执行装入操作。

如果因超过了警告数的阈值而停止了装入操作，那么可以采用 RESTART 方式开始执行另一个装入操作。装入操作将自动从最后一个一致点继续执行。或者，可以采用 REPLACE 方式启动另一个装入操作，并从输入文件开头开始装入。

iHoldQuiesce

输入。这是一个标志，如果实用程序在执行装入操作后要让表处于停顿的互斥状态，那么将此选项的值设置为 TRUE；否则，设置为 FALSE。

iCpuParallelism

输入。当构建表对象时，LOAD 实用程序为解析、转换和格式化记录而创建的进程数或线程数。此参数旨在利用分区内并行性。当装入已预先排序的数据时，此参数特别有用，这是因为将保留源数据中的记录顺序。如果此参数的值为零，那么 LOAD 实用程序在运行时将使用智能缺省值。注意：如果对包含 LOB 或 LONG VARCHAR 字段的表使用此参数，那么无论系统 CPU 的个数或者用户指定的值是多少，此参数的值都将为 1。

iDiskParallelism

输入。LOAD 实用程序为了将数据写入表空间容器而创建的进程数或线程数。如果未指定值，那么该实用程序将根据表空间容器数和表的特征来选择智能缺省值。

iNonrecoverable

输入。如果要将装入事务标记为不可恢复，并且后续前滚操作不能将其恢复，那么将此参数设置为 SQLU_NON_RECOVERABLE_LOAD。Rollforward 实用程序将跳过该事务，并将装入数据的表标记为“无效”。该实用程序还将忽略该对表执行的任何后续事务。在完成前滚操作后，只能删除这样的表。如果指定了此选项，在装入操作完成后就不会将表空间置于“备份暂挂”状态，并且在装入操作执行期间不必创建所装入数据的副本。如果要将装入事务标记为可恢复，那么将此参数设置为 SQLU_RECOVERABLE_LOAD。

iIndexingMode

输入。指定建立索引方式。有效值（是在 sqlutil 头文件中定义的，该文件位于包含目录中）是:

SQLU_INX_AUTOSELECT
LOAD 实用程序选择 REBUILD 或 INCREMENTAL 建立索引方式。

SQLU_INX_REBUILD
重建表索引。

SQLU_INX_INCREMENTAL
扩展现有索引。

SQLU_INX_DEFERRED
不更新表索引。

iAccessLevel

输入。指定访问级别。有效值为:

SQLU_ALLOW_NO_ACCESS
指定 LOAD 实用程序将互斥锁定表。

SQLU_ALLOW_READ_ACCESS
指定在装入过程中阅读器应当仍然能够看见表中的原始数据（非增量部分）。此选项仅对于装入追加（例如，装入插入）有效，而对于装入替换将忽略此选项。

iLockWithForce
输入。一个布尔标志。如果设置为 TRUE，那么 LOAD 实用程序将根据需要强制其他应用程序确保立即获得表锁定。此选项与 FORCE APPLICATIONS 命令需要相同的权限（SYSADM 或 SYSCTRL）。

在装入操作开始时，SQLU_ALLOW_NO_ACCESS 装入可能会强制互相冲突的应用程序。开始装入时，LOAD 实用程序可能会强制要尝试查询或修改表的应用程序。

在装入操作开始或结束时，SQLU_ALLOW_READ_ACCESS 装入可能会强制互相冲突的应用程序。开始装入时，LOAD 实用程序可能会强制要尝试修改表的应用程序。装入结束时，LOAD 实用程序可能会强制要尝试查询或修改表的应用程序。

iCheckPending
从版本 9.1 开始，不推荐使用此参数。请改用 iSetIntegrityPending 参数。

iRestartphase
输入。保留参数。有效值是单个空格字符 ' '。

iStatsOpt
输入。要收集的统计信息的详细程度。有效值为:

SQLU_STATS_NONE
不收集统计信息。

SQLU_STATS_USE_PROFILE
根据为当前表定义的概要文件来收集统计信息。必须使用 RUNSTATS 命令来创建此概要文件。如果当前表不存在任何概要文件，那么会返回警告，并且不会收集统计信息。

iSetIntegrityPending
输入。指定此参数以将表置于"设置完整性暂挂"状态。如果指定的值为 SQLU_SI_PENDING_CASCADE_IMMEDIATE，那么会立即将"设置完整性暂挂"状态级联至所有从属表和派生表。如果指定的值为 SQLU_SI_PENDING_CASCADE_DEFERRED，那么在检查目标表是否违反完整性前，会延迟将"设置完整性暂挂"状态级联至从属表。如果未指定此选项，那么 SQLU_SI_PENDING_CASCADE_DEFERRED 为缺省值。

piSourceUserExit
输入。指向 db2LoadUserExit 结构的指针。

piXmlParse
输入。应该对 XML 文档执行的解析类型。在包含目录中的 db2ApiDf 头文件中找到的有效值为:

DB2DMU_XMLPARSE_PRESERVE_WS
应该保留空格。

DB2DMU_XMLPARSE_STRIP_WS
应该去掉空格。

piXmlValidate
输入。指向 db2DMUXmlValidate 结构的指针。指示应该对 XML 文档执行 XML 模式验证。

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                        iUsing;      /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2LoadOut 数据结构参数

oRowsRead
输出。在执行装入操作期间读取的记录数。

oRowsSkipped
输出。在开始执行装入操作前已跳过的记录数。

oRowsLoaded
输出。已装入到目标表中的行数。

oRowsRejected
输出。不能装入的记录数。

oRowsDeleted
输出。已删除的重复行数。

oRowsCommitted
输出。已处理的记录总数：已成功装入并落实到数据库的记录数加上已跳过和已拒绝的记录数。

db2PartLoadIn 数据结构参数

piHostname
输入。用于 iFileTransferCmd 参数的主机名。如果此参数的值为 NULL，那么会将主机名缺省设置为“nohost”。不推荐使用此参数。

piFileTransferCmd
输入。文件传输命令参数。如果不需要此参数，那么必须将它设置为 NULL。不推荐使用此参数。请改用 piSourceUserExit 参数。

piPartFileLocation
输入。在 PARTITION_ONLY、LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式下，可以使用此参数来指定分布式文件的位置。在 piOutputNodes 选项指定的每个数据库分区上，都必须存在此位置。

对于 SQL_CURSOR 文件类型，此参数不能为 NULL，并且位置引用的不是路径，而是标准文件名。对于 PARTITION_ONLY 方式，这将是在每个输出数据库分区上创建的分区文件的标准基本文件名，而对于 LOAD_ONLY 方式，这将是要从每个数据库分区中读取的文件的位置。对于 PARTITION_ONLY 方式，

如果目标表包含 LOB 列，那么可以使用指定的基本名称来创建多个文件。对于 SQL_CURSOR 之外的文件类型，如果此参数的值为 NULL，那么它将缺省设置为当前目录。

piOutputNodes

输入。装入输出数据库分区的列表。如果为 NULL，那么表示定义了目标表的所有节点。

piPartitioningNodes

输入。分区节点的列表。如果为 NULL，那么表示缺省值。

piMode

输入。指定分区数据库的装入方式。有效值（是在 db2ApiDf 头文件中定义的，该文件位于包含目录中）是:

- **DB2LOAD_PARTITION_AND_LOAD**

  对数据进行分布（有可能以并行方式进行分布），并且同时在各个相应数据库分区上装入数据。

- **DB2LOAD_PARTITION_ONLY**

  对数据进行分布（有可能以并行方式进行分布），并将输出写入每个装入数据库分区上指定位置中的文件。对于 SQL_CURSOR 之外的文件类型，每个数据库分区上输出文件的名称的格式为 filename.xxx，其中 filename 是 piSourceList 指定的第一个输入文件的名称，xxx 是数据库分区号。对于 SQL_CURSOR 文件类型，每个数据库分区上输出文件的名称将由 piPartFileLocation 参数来确定。有关如何指定每个数据库分区上数据库分区文件的位置的信息，请参阅 piPartFileLocation 参数。

  注: 此方式不能用于 CLI LOAD。

DB2LOAD_LOAD_ONLY

假定已对数据进行分布；将跳过分布过程，并且在相应的数据库分区上同时装入数据。对于不是 SQL_CURSOR的文件类型，每个数据库分区上的输入文件名的格式应为 filename.xxx，其中 filename 是 piSourceList 指定的第一个文件的名称，xxx 是一个 13 位的数据库分区号。对于 SQL_CURSOR 文件类型，每个数据库分区上输入文件的名称将由 piPartFileLocation 参数来确定。有关如何指定每个数据库分区上数据库分区文件的位置的信息，请参阅 piPartFileLocation 参数。

注: 当装入位于远程客户机上的数据文件时，不能使用此方式；也不能将此方式用于 CLI LOAD。

DB2LOAD_LOAD_ONLY_VERIFY_PART

假定已对数据进行分布，但是数据文件不包含数据库分区头。将跳过分布过程，并且在相应的数据库分区上同时装入数据。在装入操作期间，将检查每一行以验证它是否在正确的数据库分区中。如果指定了文件类型修饰符 dumpfile，就会将发生数据库分区违例的行放到转储文件中。否则会删除那些行。如果在特定装入数据库分区上发生数据库分区违例，就会将一条有关该数据库分区的警告写入装入消息文件。期望每个数据库分区上的输入文件名的格式为 filename.xxx，其中 filename 是 piSourceList 指定的第一个文件的名称，xxx 是一个 13 位的数据库分区号。

注: 当装入位于远程客户机上的数据文件时，不能使用此方式；也不能将此方式用于 CLI LOAD。

DB2LOAD_ANALYZE
生成最佳分布图（在所有数据库分区之间均匀地分布数据）。

piMaxNumPartAgents
输入。分区代理程序的最大数目。如果为 NULL，那么表示使用缺省值 25。

piIsolatePartErrs
输入。指示装入操作将如何对各个数据库分区上发生的错误作出反应。有效值（是在 db2ApiDf 头文件中定义的，该文件位于包含目录中）是:

DB2LOAD_SETUP_ERRS_ONLY
在此方式下，设置期间在数据库分区上发生的错误（例如，访问数据库分区时发生的问题，或者访问数据库分区上的表空间或表时发生的问题）将导致装入操作在发生故障的数据库分区上停止运行，但是在其余数据库分区上将继续运行。装入数据时，数据库分区上发生的错误将导致整个操作失败，并且将回滚到每个数据库分区上的最后一个一致点。

DB2LOAD_LOAD_ERRS_ONLY
在此方式下，设置期间在数据库分区上发生的错误将导致整个装入操作失败。如果在装入数据时发生错误，那么会将出错的数据库分区回滚到它们的最后一个一致点。装入操作将在其余数据库分区上继续运行，直到发生故障或者装入了所有数据为止。在装入了所有数据的数据库分区上，在完成装入操作后，这些数据将不可视。由于其他数据库分区中发生了错误，因此事务将异常中止。在执行“重新启动装入”操作前，所有数据库分区上的数据都将保持不可视状态。这将使得新装入的数据在完成了装入操作的数据库分区上可视，并且在产生错误的数据库分区上将继续执行装入操作。

注: 当 iAccessLevel 设置为 SQLU_ALLOW_READ_ACCESS 并且还指定了复制目标时，将不能使用此方式。

DB2LOAD_SETUP_AND_LOAD_ERRS
在此方式下，设置期间或装入数据期间发生的数据库分区级别错误将导致仅在受影响的数据库分区上停止处理装入操作。与 DB2LOAD_LOAD_ERRS_ONLY 方式一样，如果在装入数据时发生了数据库分区错误，那么在执行“重新启动装入”操作前，所有数据库分区上的数据都将保持不可视状态。

注: 当 iAccessLevel 设置为 SQLU_ALLOW_READ_ACCESS 并且还指定了复制目标时，将不能使用此方式。

DB2LOAD_NO_ISOLATION
在执行装入操作期间发生的任何错误都会导致事务异常中止。如果此参数为 NULL，那么它将缺省设置为 DB2LOAD_LOAD_ERRS_ONLY（除非 iAccessLevel 设置为 SQLU_ALLOW_READ_ACCESS 并且还指定了复制目标。在这种情况下，缺省值为 DB2LOAD_NO_ISOLATION）。

piStatusInterval

输入。指定在生成进度消息前要装入的数据的兆字节（MB）数。有效值是 1 到 4000 范围内的整数。如果指定了 NULL，那么将使用缺省值 100。

piPortRange

输入。用于内部通信的 TCP 端口范围。如果为 NULL，那么将使用的端口范围是 6000 到 6063。

piCheckTruncation

输入。使 LOAD 实用程序在进行输入/输出时检查记录是否被截断。有效值为 TRUE 或 FALSE。如果为 NULL，那么缺省值为 FALSE。

piMapFileInput

输入。分布图输入文件名。如果方式不是 ANALYZE，那么应将此参数设置为 NULL。如果方式是 ANALYZE，那么必须指定此参数。

piMapFileOutput

输入。分布图输出文件名。用于 piMapFileInput 的规则在此处也适用。

piTrace

输入。当您需要复查所有数据转换过程的转储和散列值的输出时，指定要跟踪的记录数。如果为 NULL，那么记录数缺省设置为 0。

piNewline

输入。如果还指定了文件类型修饰符 RECLEN，那么会强制 LOAD 实用程序检查 ASC 数据记录的末尾是否具有换行符。它的值可能是 TRUE 或 FALSE。如果为 NULL，那么值缺省设置为 FALSE。

piDistfile

输入。数据库分区分发文件的名称。如果指定了 NULL，那么值缺省设置为 “DISTFILE”。

piOmitHeader

输入。当使用 DB2LOAD_PARTITION_ONLY 方式时，指示不应将分布图头包含在数据库分区文件中。它的值可能是 TRUE 或 FALSE。如果为 NULL，那么缺省值为 FALSE。

piRunStatDBPartNum

指定要收集统计信息的数据库分区。缺省值是输出数据库分区列表中的第一个数据库分区。

db2LoadNodeList 数据结构参数

piNodeList

输入。节点号组成的数组。

iNumNodes

输入。piNodeList 数组中的节点数。如果为 0，那么表示缺省值（即，定义了目标表的所有节点）。

db2LoadPortRange 数据结构参数

iPortMin

输入。较小的端口号。

iPortMax

输入。较大的端口号。

db2PartLoadOut 数据结构参数

oRowsRdPartAgents

输出。所有分区代理程序已读取的总行数。

oRowsRejPartAgents

输出。所有分区代理程序已拒绝的总行数。

oRowsPartitioned

输出。所有分区代理程序已分区的总行数。

poAgentInfoList

输出。在执行装入到分区数据库的装入操作期间，可能涉及到下列装入处理实体：装入代理程序、分区代理程序、预分区代理程序、文件传输命令代理程序和 Load-To-File 代理程序（在《数据移动实用程序指南和参考》中描述了这些代理程序）。poAgentInfoList 输出参数用于返回有关参与装入操作的每个装入代理程序的调用者信息。该列表中的每个条目都包含以下信息:

oAgentType

这是一个标记，用来指示该条目描述哪种类型的装入代理程序。

oNodeNum

代理程序执行操作所在的数据库分区的编号。

oSqlcode

代理程序处理产生的最终 sqlcode。

oTableState

执行了代理程序的数据库分区上表的最终状态（仅与装入代理程序有关）。

在调用 API 前，由该 API 的调用者来为此列表分配内存。调用者还应指出他们在 iMaxAgentInfoEntries 参数中为其分配了内存的条目数。如果调用者将 poAgentInfoList 设置为 NULL 或者将 iMaxAgentInfoEntries 设置为 0，那么不会返回有关装入代理程序的任何信息。

iMaxAgentInfoEntries

输入。用户为 poAgentInfoList 分配的代理程序信息条目的最大数目。通常，将此参数设置为装入操作中涉及到的数据库分区数的 3 倍就已足够。

oNumAgentInfoEntries

输出。由装入操作生成的代理程序信息条目的实际数目。只要 iMaxAgentInfoEntries 大于或等于 oNumAgentInfoEntries，就会将此条目数返回给 poAgentInfoList 参数中的用户。如果 iMaxAgentInfoEntries 小于 oNumAgentInfoEntries，那么 poAgentInfoList 中返回的条目数等于 iMaxAgentInfoEntries。

db2LoadAgentInfo 数据结构参数

oSqlcode

输出。代理程序处理产生的最终 sqlcode。

oTableState

输出。此输出参数的作用不是在执行装入操作后报告表可能处于的每种状态。而是只报告表可能处于的一少部分状态，以便让调用者大致了解在装入处理期间对表执行了哪些操作。此值仅与装入代理程序有关。可能的值为:

DB2LOADQUERY_NORMAL

指示在数据库分区上已成功完成了装入，并且表已脱离“正在装入”（或者“装入暂挂”）状态。在这种情况下，由于需要进一步进行约束处理，因此，表可能仍然处于“设置完整性暂挂”状态。但是并不会报告这种情况，因为这是正常的。

DB2LOADQUERY_UNCHANGED

指示因发生错误而导致装入作业异常中止处理，但是尚未更改表在数据库分区上所处的状态，表仍然处于在调用 db2Load 前的状态。不需要对这样的数据库分区执行“重新启动装入”或“终止装入”操作。

DB2LOADQUERY_LOADPENDING

指示装入作业在处理期间异常中止，但是数据库分区上的表仍然处于装入暂挂状态，这种状态表示必须终止或重新启动该数据库分区上的装入作业。

oNodeNum

输出。代理程序执行操作所在的数据库分区的编号。

oAgentType

输出。代理程序类型。有效值（是在 db2ApiDf 头文件中定义的，该文件位于包含目录中）是:

- DB2LOAD_LOAD_AGENT
- DB2LOAD_PARTITIONING_AGENT
- DB2LOAD_PRE_PARTITIONING_AGENT
- DB2LOAD_FILE_TRANSFER_AGENT
- DB2LOAD_LOAD_TO_FILE_AGENT

特定于 db2gLoadStruct 数据结构的参数

iFileTypeLen

输入。指定 iFileType 参数的长度（以字节计）。

iLocalMsgFileLen

输入。指定 iLocalMsgFileName 参数的长度（以字节计）。

iTempFilesPathLen

输入。指定 iTempFilesPath 参数的长度（以字节计）。

piXmlPathList

输入。指向 sqlu_media_list 的指针，它的 media_type 字段设置为 SQLU_LOCAL_MEDIA，并且它的 sqlu_media_entry 结构列示客户机上用于存储 xml 文件的路径。

特定于 db2gLoadIn 数据结构的参数

iUseTablespaceLen

输入。piUseTablespace 参数的长度（以字节计）。

piXmlParse

输入。应该对 XML 文档执行的解析类型。在包含目录中的 db2ApiDf 头文件中找到的有效值为:

DB2DMU_XMLPARSE_PRESERVE_WS

应该保留空格。

DB2DMU_XMLPARSE_STRIP_WS
应该去掉空格。

piXmlValidate
输入。指向 db2DMUXmlValidate 结构的指针。指示应该对 XML 文档执行 XML 模式验证。

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                       iUsing;       /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

特定于 db2gPartLoadIn 数据结构的参数

piReserved1
保留以备将来使用。

iHostnameLen
输入。piHostname 参数的长度（以字节计）。

iFileTransferLen
输入。piFileTransferCmd 参数的长度（以字节计）。

iPartFileLocLen
输入。piPartFileLocation 参数的长度（以字节计）。

iMapFileInputLen
输入。piMapFileInput 参数的长度（以字节计）。

iMapFileOutputLen
输入。piMapFileOutput 参数的长度（以字节计）。

iDistfileLen
输入。piDistfile 参数的长度（以字节计）。

使用说明

数据的装入顺序与在输入文件中的出现顺序相同。如果需要使用特定顺序，那么在尝试装入前应该对数据进行排序。

LOAD 实用程序将根据现有定义来构建索引。异常表用来处理重复的唯一键。该实用程序不会强制实施引用完整性、执行约束检查或者更新依赖于正在装入的表的总结表。包含参考约束或检查约束的表将处于“设置完整性暂挂”状态。使用 REFRESH IMMEDIATE 定义的、并且依赖于正在装入的表的总结表也将处于“设置完整性暂挂”状态。发出 SET INTEGRITY 语句以使表脱离“设置完整性暂挂”状态。不能对重复的总结表执行装入操作。

对于集群索引，应该在执行装入前按集群索引对数据进行排序。当装入到多维集群（MDC）表时，不需要对数据进行排序。

装入会话 - CLP 示例

示例 1

TABLE1 有 5 列:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 有 6 个元素:

- ELE1 位置 01 到 20
- ELE2 位置 21 到 22
- ELE3 位置 23 到 23
- ELE4 位置 24 到 27
- ELE5 位置 28 到 31
- ELE6 位置 32 到 32
- ELE7 位置 33 到 40

数据记录:

```
1...5...10...15...20...25...30...35...40
   Test data 1         XXN 123abcdN
   Test data 2 and 3   QQY    XXN
Test data 4,5 and 6 WWN6789    Y
```

以下命令从文件中装入表:

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
         method L (1 20, 21 22, 24 27, 28 31)
         null indicators (0,0,23,32)
         insert into table1 (col1, col5, col2, col3)
```

注:

1. 由于在 MODIFIED BY 参数中指定了 striptblanks，所以将强制截断 VARCHAR 列中的空白（例如，第 1、2 和 3 行中的 COL1 列长度分别为 11、17 和 19 个字节）。
2. 在 MODIFIED BY 参数中指定了 reclen=40，这表示在每个输入记录末尾没有换行符，并且每个记录的长度为 40 个字节。装入该表时，不会使用最后 8 个字节。
3. 由于在输入文件中未提供 COL4，所以将使用该列的缺省值来将其插入到 TABLE1 中（定义该列时指定了 NOT NULL WITH DEFAULT）。
4. 位置 23 和 32 用来指示: 装入 TABLE1 时，对于给定的行，是否将 COL2 和 COL3 设置为 NULL。对于给定记录，如果该列的空指示符位置包含 Y，那么该列将是 NULL。如果包含 N，那么将输入记录（由 L(.....) 定义）中该列的数据位置中包含的数据值用作该行的列数据源。在此示例中，第 1 行中的任何一列都不是 NULL；第 2 行中的 COL2 是 NULL；第 3 行中的 COL3 是 NULL。
5. 在此示例中，将 COL1 和 COL5 的 NULL INDICATORS 指定为 0（零），表示数据不可为空。

6. 给定列的 NULL INDICATOR 可以在输入记录中的任何位置，但必须指定该位置，并且必须提供 Y 或 N 值。

示例 2（使用转储文件）

表 FRIENDS 的定义如下所示:

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

如果尝试将下列数据记录装入到表中:

```
    23, 24, bobby
    , 45, john
    4,, mary
```

将拒绝第二行，这是因为第一个 INT 是 NULL，但列定义指定了 NOT NULL。包含与 DEL 格式不一致的初始字符的列将生成错误，将拒绝该记录。您可以将此类记录写入转储文件。

在一列中，将忽略字符定界符外部的 DEL 数据，但会生成警告。例如:

```
22,34,"bob"
   24,55,"sam" sdf
```

load 实用程序将在表的第三列中装入 ″sam″，并且将会在一条警告消息中标记字符 ″sdf″。不会拒绝该记录。另一个示例:

```
    22 3, 34,"bob"
```

load 实用程序将装入 22,34,"bob" 并生成一条警告消息，该消息指出忽略了第一列中 22 后面的某些数据。不会拒绝该记录。

示例 3（装入包含标识列的表）

TABLE1 有 4 列:

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 与 TABLE1 相同，但 C2 是 GENERATED ALWAYS 标识列。

DATAFILE1 中的数据记录（DEL 格式）:

```
"Liszt"
   "Hummel",,187.43, H
   "Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

DATAFILE2 中的数据记录（DEL 格式）:

```
"Liszt", 74.49, A
   "Hummel", 0.01, H
   "Grieg", 66.34, G
"Satie", 818.23, I
```

注:

1. 由于在 DATAFILE1 中没有为第 1 行和第 2 行提供标识值，因此以下命令将为这两行生成标识值。但是，分别对第 3 行和第 4 行指定了用户提供的标识值 100 和 101。

```
db2 load from datafile1.del of del replace into table1
```

2. 要将 DATAFILE1 装入到 TABLE1 中，以便为所有行生成标识值，请发出下列其中一个命令:

```
db2 load from datafile1.del of del method P(1, 3, 4)
        replace into table1 (c1, c3, c4)
   db2load from datafile1.del of del modified by identityignore
        replace into table1
```

3. 要将 DATAFILE2 装入到 TABLE1 中，以便为每一行生成标识值，请发出下列其中一个命令:

```
db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
   db2 load from datafile2.del of del modified by identitymissing
        replace into table1
```

4. 要将 DATAFILE1 装入到 TABLE2 中以便对第 3 行和第 4 行指定标识值 100 和 101，请发出以下命令:

```
db2 load from datafile1.del of del modified by identityoverride
        replace into table2
```

   在本例中，由于指示 load 实用程序使用用户提供的值来覆盖系统生成的标识值，所以将拒绝第 1 行和第 2 行。但是，如果用户未提供值，就必须拒绝该行，这是因为标识列在隐式情况下不能为 NULL。

5. 如果将 DATAFILE1 装入到 TABLE2 中，但未使用任何与标识相关的文件类型修饰符，那么将装入第 1 行和第 2 行，但将拒绝第 3 行和第 4 行，这是因为这两行提供了它们自己的非空值，而标识列是 GENERATED ALWAYS 列。

示例 4（从 CURSOR 装入）

MY.TABLE1 有 3 列:

- ONE INT
- TWO CHAR(10)
- THREE DATE

MY.TABLE2 有 3 列:

- ONE INT
- TWO CHAR(10)
- THREE DATE

游标 MYCURSOR 是按以下方式定义的:

```
declare mycursor cursor for select * from my.table1
```

以下命令将 MY.TABLE1 中的所有数据装入到 MY.TABLE2 中:

```
load from mycursor of cursor method P(1,2,3) insert into
    my.table2(one,two,three)
```

注:

1. 在单个 LOAD 命令中只指定了一个游标名。即，不允许 `load from mycurs1, mycurs2 of cursor...`。

2. 对于从游标装入来说，有效的 METHOD 值只有 P 和 N。
3. 在此示例中，由于插入列列表 (one,two,three) 表示缺省值，所以可以将 METHOD P 和该插入列列表省略。
4. MY.TABLE1 可以是表、视图、别名或昵称。

SET INTEGRITY

SET INTEGRITY 语句用于:

- 通过对那些表执行必需的完整性处理来使一个或多个表脱离“设置完整性暂挂”状态（先前称为“检查暂挂状态”）。
- 在未对那些表执行必需的完整性处理的情况下使一个或多个表脱离“设置完整性暂挂”状态。
- 将一个或多个表置于“设置完整性暂挂”状态。
- 将一个或多个表置于完全访问状态。
- 修剪一个或多个登台表的内容。

使用该语句对已装入或连接的表执行完整性处理时，系统可通过仅检查追加的部分是否违反约束来对该表执行增量处理。如果主题表是具体化查询表或登台表，且对其基础表执行了装入、连接或拆离操作，那么系统可通过增量方式刷新具体化查询表或仅将其基础表的变化量部分以增量方式传播至登台表。但是，在某些情况下，系统将无法执行此类优化并且将改为执行完全的完整性处理以确保数据完整性。通过检查整个表是否违反约束、重新计算具体化查询表的定义或者将登台表标记为不一致来执行完全的完整性处理。后者意味着需要对其关联的具体化查询表进行完全刷新。另外，还存在一种情况: 您可能需要通过指定 INCREMENTAL 选项来显式地请求增量处理。

SET INTEGRITY 语句处于事务控制下。

调用

可以将此语句嵌入应用程序中，还可通过使用动态 SQL 语句来发出该语句。它是一个可执行语句，仅当 DYNAMICRULES 运行行为对于程序包有效时才能动态编译该语句（SQLSTATE 42509）。

权限

执行 SET INTEGRITY 语句时所需的特权取决于用途，如下所述。

- 使表脱离“设置完整性暂挂”状态并执行必需的完整性处理。

  语句授权标识拥有的特权必须至少包括下列其中一项权限或特权:
  - 对下列表的 CONTROL 特权:
    - 执行完整性处理的表，如果为一个或多个这种表提供了异常表，那么还需要对异常表的 INSERT 特权
    - 通过语句隐式设置为“设置完整性暂挂”状态的所有派生外键表、派生立即具体化查询表和派生立即登台表
  - LOAD 权限（具有条件）。必须先满足下列所有条件，才能将 LOAD 权限视为提供有效的特权:
    - 必需的完整性处理不涉及以下操作:

- 刷新具体化查询表
- 传播至登台表
- 更新生成列或标识列

- 如果为一个或多个表提供了异常表，那么将在完整性处理期间授予对执行了完整性处理的表和相关联的异常表的必需访问权。也就是:
  - 对执行了完整性处理的每个表的 SELECT 和 DELETE 特权，以及
  - 对异常表的 INSERT 特权

– SYSADM 或 DBADM 权限

- 在不执行必需的完整性处理的情况下使表脱离“设置完整性暂挂”状态。

  语句授权标识拥有的特权必须至少包括下列其中一项权限或特权:

  – 对所处理的表的 CONTROL 特权；对那些通过语句隐式设置为“设置完整性暂挂”状态的所有派生外键表、派生立即具体化查询表和派生立即登台表的 CONTROL 特权
  – LOAD 权限
  – SYSADM 或 DBADM 权限
- 将表置于“设置完整性暂挂”状态。

  语句授权标识拥有的特权必须至少包括下列其中一项权限或特权:

  – 对下列表的 CONTROL 特权:
    - 所指定的表
    - 将由语句置于“设置完整性暂挂”状态的派生外键表
    - 将由语句置于“设置完整性暂挂”状态的派生立即具体化查询表
    - 将由语句置于“设置完整性暂挂”状态的派生立即登台表
  – LOAD 权限
  – SYSADM 或 DBADM 权限
- 将表置于完全访问状态。

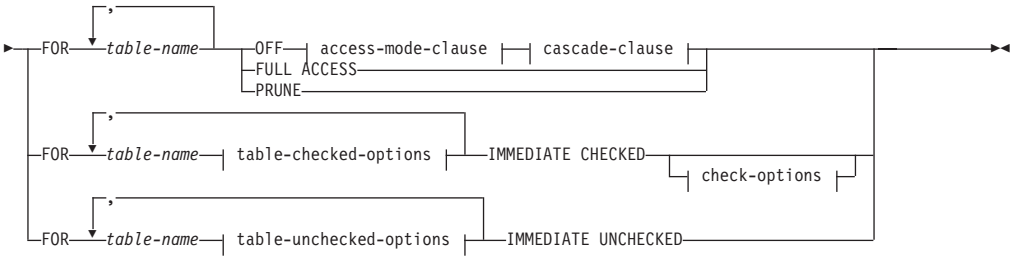
  语句授权标识拥有的特权必须至少包括下列其中一项权限或特权:

  – 对已置于完全访问状态的表的 CONTROL 特权
  – LOAD 权限
  – SYSADM 或 DBADM 权限
- 修剪登台表。

  语句授权标识拥有的特权必须至少包括下列其中一项权限或特权:

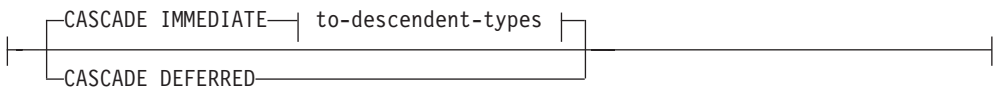
  – 对所修剪表的 CONTROL 特权
  – SYSADM 或 DBADM 权限

语法



access-mode-clause:

cascade-clause:



to-descendent-types:

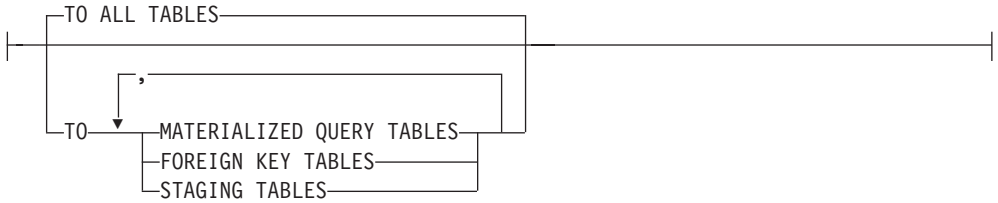
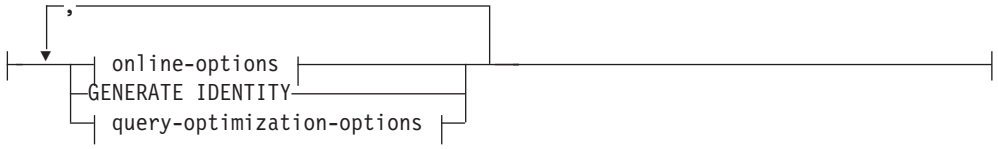


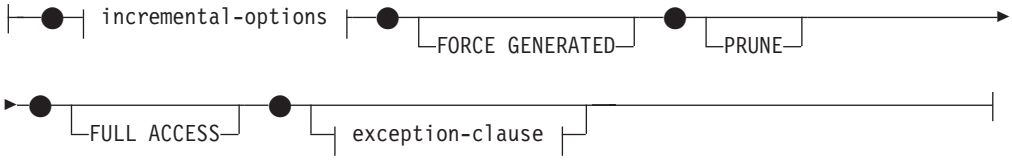
table-checked-options:



online-options:

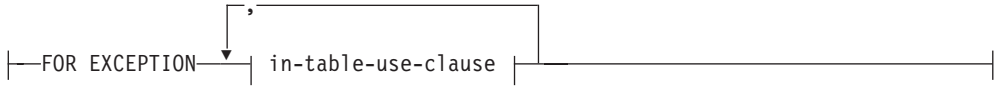
query-optimization-options:

check-options:



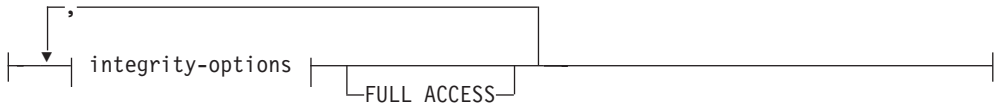
incremental-options:

exception-clause:

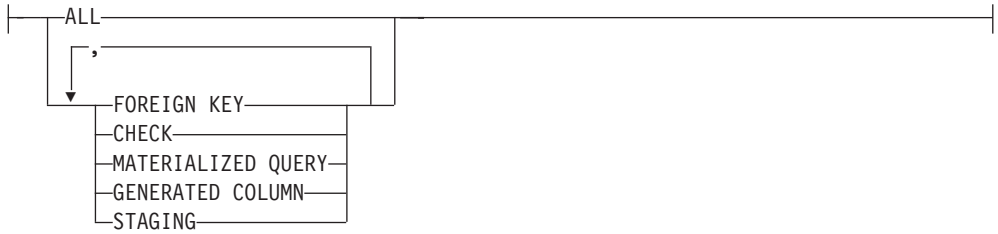


in-table-use-clause:

table-unchecked-options:



integrity-options:



描述

FOR *table-name*

标识一个或多个要进行完整性处理的表。它必须是一个在目录中描述的表且不能是视图、目录表或类型表。

OFF

指定将表置于“设置完整性暂挂”状态。仅允许对处于“设置完整性暂挂”状态的表执行非常有限的活动。

access-mode-clause

指定处于“设置完整性暂挂”状态的表的可读性。

NO ACCESS

指定将表置于“设置完整性暂挂无访问”状态，这不允许对表进行读或写访问。

READ ACCESS

指定将表置于“设置完整性暂挂读访问”状态，这允许对表的非追加部分进行读访问。不允许对处于“设置完整性暂挂无访问”状态的表指定此选项（SQLSTATE 428FH）。

cascade-clause

指定是否立即将 SET INTEGRITY 语句中引用的表的“设置完整性暂挂”状态级联至派生表。

CASCADE IMMEDIATE

指定是否立即将“设置完整性暂挂”状态扩展至派生表。

to-descendent-types

指定“设置完整性暂挂”状态将立即级联至的派生表类型。

TO ALL TABLES

指定将“设置完整性暂挂”状态立即级联至调用列表中表的所有派生表。派生表包括所有派生外键表、立即登台表以及立即具体化查询表（调用列表中表的后代或者派生外键表的后代）。

指定 TO ALL TABLES 等效于在同一语句中同时指定 TO FOREIGN KEY TABLES、TO MATERIALIZED QUERY TABLES 和 TO STAGING TABLES。

TO MATERIALIZED QUERY TABLES

只有指定了 TO MATERIALIZED QUERY TABLES 时，才会立即将“设置完整性暂挂”状态级联至派生立即具体化查询表。稍后使表脱离“设置完整性暂挂”状态时，如果必要，那么会将其他派生表置于“设置完整性暂挂”状态。如果同时指定了 TO FOREIGN KEY TABLES 和 TO MATERIALIZED QUERY TABLES，那么会立即将“设置完整性暂挂”状态级联至所有派生外键表、调用列表中表的所有派生立即具体化查询表，以及级联至所有立即具体化查询表（派生外键表的后代）。

TO FOREIGN KEY TABLES

指定是否立即将“设置完整性暂挂”状态级联至派生外键表。稍后使表脱离“设置完整性暂挂”状态时，如果必要，那么会将其他派生表置于“设置完整性暂挂”状态。

TO STAGING TABLES

指定是否立即将“设置完整性暂挂”状态级联至派生登台表。稍后使表脱离“设置完整性暂挂”状态时，如果必要，那么会将其他派生表置于“设置完整性暂挂”状态。如果同时指定了 TO FOREIGN KEY TABLES 和 TO STAGING TABLES，那么会立即将“设置完整性暂挂”状态级联至所有派生外键表、调用列表中表的所有派生立即登台表，以及级联至所有立即登台表（派生外键表的后代）。

CASCADE DEFERRED

指定仅将调用列表中的表置于“设置完整性暂挂”状态。派生表的状态将保持不变。稍后检查派生外键表的父表是否违反约束时，可能将派生外键表隐式地置

于“设置完整性暂挂”状态。检查派生立即具体化查询表和派生立即登台表的某一个基础表是否存在完整性违例时，可能将派生立即具体化查询表和派生立即登台表隐式地置于“设置完整性暂挂”状态。

如果未指定 *cascade-clause*，那么会立即将“设置完整性暂挂”状态级联至所有派生表。

IMMEDIATE CHECKED

指定通过对表执行必需的完整性处理来使表脱离“设置完整性暂挂”状态。这通过遵循在 SYSCAT.TABLES 目录视图中设置的 STATUS 和 CONST_CHECKED 列中的信息来实现。也就是:

- STATUS 列中的值必须是“C”（表处于“设置完整性暂挂”状态），否则将返回错误（SQLSTATE 51027），除非表是派生外键表、派生具体化查询表，或者在列表中指定的表的派生登台表处于“设置完整性暂挂”状态并且该派生登台表的中间祖代也在此列表中。
- 如果所检查的表处于“设置完整性暂挂”状态，那么 CONST_CHECKED 中的值指示要检查的完整性选项。

使表脱离“设置完整性暂挂”状态时，如果必要，那么会将该表的派生表置于“设置完整性暂挂”状态。将返回警告，指示已将派生表置于“设置完整性暂挂”状态（SQLSTATE 01586）。

如果表是系统维护的具体化查询表，那么将根据需要针对查询进行检查并刷新数据。（IMMEDIATE CHECKED 不能用于用户维护的具体化查询表。）如果表是登台表，那么将根据需要针对其查询定义进行检查并传播数据。

检查子表的完整性时:

- 其父表不能处于“设置完整性暂挂”状态，或者
- 必须在同一 SET INTEGRITY 语句中检查其各个父表是否违反约束

刷新立即具体化查询表或将变化量传播至登台表时:

- 其基础表不能处于“设置完整性暂挂”状态，或者
- 必须在同一 SET INTEGRITY 语句中检查其各个基础表

否则，将返回错误（SQLSTATE 428A8）。

table-checked-options

online-options

指定正被处理的表的可访问性。

ALLOW NO ACCESS

指定任何其他用户均不能访问正被处理的表，除非用户使用“未落实的读”隔离级别。

ALLOW READ ACCESS

指定其他用户对正被处理的表具有只读访问权。

ALLOW WRITE ACCESS

指定其他用户对正被处理的表具有读写访问权。

GENERATE IDENTITY

指定如果表包含标识列，那么将由 SET INTEGRITY 语句生成值。缺省情况下，指定 GENERATE IDENTITY 选项时，仅连接的行将通过 SET INTEGRITY 语句生成它们的标识列值。必须将 NOT INCREMENTAL 选

项与 GENERATE IDENTITY 选项一起指定，以便让 SET INTEGRITY 语句为表中的所有行（包括连接的行、装入的行和现有行）生成标识列值。如果未指定 GENERATE IDENTITY 选项，那么表中所有行的当前标识列值将保持不变。

query-optimization-options

指定查询优化选项以进行 REFRESH DEFERRED 具体化查询表维护。

ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY

指定将 CURRENT REFRESH AGE 专用寄存器设置为“ANY”时，对 *table-name* 的维护将允许使用 REFRESH DEFERRED 具体化查询表来优化对 *table-name* 进行维护的表。如果 *table-name* 不是 REFRESH DEFERRED 具体化查询表，那么将返回错误（SQLSTATE 428FH）。在查询优化期间始终使用 REFRESH IMMEDIATE 具体化查询表。

check-options

incremental-options

INCREMENTAL

指定对表的追加部分（如果有）应用完整性处理。如果不能满足此类请求（即，系统检测到需要检查整个表的数据完整性），那么将返回错误（SQLSTATE 55019）。

NOT INCREMENTAL

指定对整个表应用完整性处理。如果表是具体化查询表，那么将重新计算该具体化查询表的定义。如果表上至少定义了一种约束，那么此选项将导致对派生外键表和派生立即具体化查询表进行完全处理。如果表是登台表，那么会将其设置为不一致状态。

如果未指定 *incremental-options* 子句，那么系统将确定是否可进行增量处理；如果不能进行增量处理，那么将检查整个表。

FORCE GENERATED

如果表包含 GENERATED BY EXPRESSION 列，那么将根据表达式计算值并将其存储在列中。如果未指定此选项，那么会将当前值与表达式的计算值进行比较，如同“等同性检查”约束仍生效一样。如果以增量方式处理表的完整性，那么将仅对追加的部分计算所生成的列。

PRUNE

只能对登台表指定此选项。指定将修剪登台表的内容以及将登台表设置为不一致状态。如果 *table-name* 列表中的任何表都不是登台表，那么将返回错误（SQLSTATE 428FH）。如果还指定了“INCREMENTAL 检查”选项，那么将返回错误（SQLSTATE 428FH）。

FULL ACCESS

指定在执行 SET INTEGRITY 语句后表将成为完全可访问表。

如果以增量方式处理调用列表中的基础表（具有从属立即具体化查询表或从属立即登台表），那么在执行 SET INTEGRITY 语句后，会根据需要将基础表置于“无数据移动”状态。使所有可以增量方式刷新的从属立即具体化查询表和登台表均脱离“设置完整性暂挂”状态后，将自动使基础表脱离“无数据移动”状态并进入“完全访问”状态。如果将 FULL ACCESS 选项与 IMMEDIATE CHECKED 选项一起指定，那么会将基础表直接置于“完全访

问”状态（绕过“无数据移动”状态）。尚未刷新的从属立即具体化查询表可能会在后续 REFRESH TABLE 语句中进行完全重新计算，并且如果表的追加部分尚未传播至从属立即登台表，那么会将这些从属立即登台表标记为不一致。

如果调用列表中的基础表需要完全处理，或者没有从属立即具体化查询表或从属立即登台表，那么在执行 SET INTEGRITY 语句后，不管是否指定了 FULL ACCESS 选项，都会直接将基础表置于“完全访问”状态。

exception-clause

FOR EXCEPTION

指定将违反任何所检查约束的行移入异常表。即使检测到错误，也使表脱离“设置完整性暂挂”状态。将返回警告，指示已将一行或多行移入异常表（SQLSTATE 01603）。

如果未指定 FOR EXCEPTION 选项，并且违反了任何约束，那么将仅返回所检测到的第一个违例（SQLSTATE 23514）。如果任何表中存在违例，那么会将所有表置于“设置完整性暂挂”状态。

检查是否存在约束违例时，如果发现违例，建议始终使用 FOR EXCEPTION 选项来阻止回滚 SET INTEGRITY 语句。

IN *table-name*

指定从中移走违反约束的行的表。必须对所检查的每个表指定一个异常表。不能对具体化查询表或登台表指定此子句（SQLSTATE 428A7）。

USE *table-name*

指定要将错误行移入到的异常表。

FULL ACCESS

如果将 FULL ACCESS 选项指定为语句的唯一操作，那么会将表置于“完全访问”状态，而不重新检查是否存在完整性违例。但是，尚未刷新的从属立即具体化查询表可能会在后续 REFRESH TABLE 语句中进行完全重新计算，并且如果表的变化量部分尚未传播至从属立即登台表，那么会将这些从属立即登台表更改为“未完成”状态。只能对处于“无数据移动”状态或“无访问状态”的表指定此选项，而不能对处于“设置完整性暂挂”状态的表指定此选项（SQLSTATE 428FH）。

PRUNE

只能对登台表指定此选项。指定将修剪登台表的内容以及将登台表设置为不一致状态。如果 *table-name* 列表中的任何表都不是登台表，那么将返回错误（SQLSTATE 428FH）。

table-unchecked-options

integrity-options

用于定义使表脱离“设置完整性暂挂”状态时将绕过的必需完整性处理类型。

ALL

将立即使表脱离“设置完整性暂挂”状态，而不必执行表的任何必需完整性处理。

FOREIGN KEY

使表脱离“设置完整性暂挂”状态时，将不执行必需的外键约束检查。

CHECK

使表脱离“设置完整性暂挂”状态时，将不执行必需的“检查约束”检查。

MATERIALIZED QUERY
使表脱离“设置完整性暂挂”状态时，将不执行必需的具体化查询表刷新。

GENERATED COLUMN
使表脱离“设置完整性暂挂”状态时，将不执行必需的“生成列约束”检查。

STAGING
使表脱离“设置完整性暂挂”状态时，不执行将数据传播至登台表这一必需操作。

将特定类型的完整性处理标记为“绕过”后，如果不需要对表执行任何其他类型的完整性处理，那么立即使该表脱离“设置完整性暂挂”状态。

FULL ACCESS
指定在执行 SET INTEGRITY 语句后表将成为完全可访问表。

如果以增量方式处理调用列表中的基础表，并且该基础表具有从属立即具体化查询表或从属立即登台表，那么在执行 SET INTEGRITY 语句后，会根据需要将基础表置于“无数据移动”状态。使所有可以增量方式刷新的从属立即具体化查询表和登台表均脱离了“设置完整性暂挂”状态后，将自动使基础表脱离“无数据移动”状态并进入“完全访问”状态。如果将 FULL ACCESS 选项与 IMMEDIATE UNCHECKED 选项一起指定，那么会将基础表直接置于“完全访问”状态（绕过“无数据移动”状态）。尚未刷新的从属立即具体化查询表可能会在后续 REFRESH TABLE 语句中进行完全重新计算，并且如果表的追加部分尚未传播至从属立即登台表，那么会将这些从属立即登台表标记为不一致。

如果调用列表中的基础表需要完全处理，或者没有从属立即具体化查询表或从属立即登台表，那么在执行 SET INTEGRITY 语句后，不管是否指定了 FULL ACCESS 选项，都会直接将基础表置于“完全访问”状态。

如果已将 FULL ACCESS 选项与 IMMEDIATE UNCHECKED 选项一起指定，并且该语句没有使表脱离“设置完整性暂挂”状态，那么将返回错误（SQLSTATE 428FH）。

IMMEDIATE UNCHECKED
指定下列其中一项:

- 在未立即执行任何必需的完整性处理的情况下，使表脱离“设置完整性暂挂”状态。
- 通过后续使用 IMMEDIATE CHECKED 选项的 SET INTEGRITY 语句使表脱离“设置完整性暂挂”状态时，表将绕过一种或多种类型的必需完整性处理。

使用此选项前，请考虑此选项的数据完整性隐患。请参阅下面的“注释”部分。

注意

- 对表（处于某种与受限设置完整性相关的状态）的影响:
  - 对处于“读访问”状态或“无访问”状态的表禁止使用 INSERT、UPDATE 或 DELETE。另外，如果任一语句要求对表作出此类修改，而该表处于这样的状态，那么将拒绝该语句。例如，如果父表级联至处于“无访问”状态的从属表，那么不允许删除此父表中的行。
  - 禁止对处于“无访问”状态的表使用 SELECT 语句。另外，如果任一语句需要对表的读访问权，而该表处于“无访问”状态，那么将拒绝该语句。

- 通常，对表添加的新约束会立即生效。但是，如果表处于“设置完整性暂挂”状态，那么在使表脱离“设置完整性暂挂”状态前，将延迟对任何新约束的检查。如果表处于“设置完整性暂挂”状态，那么添加新约束时，会将表置于“设置完整性暂挂无访问”状态，这是因为在验证数据的有效性时存在风险。
- CREATE INDEX 语句不能引用任何处于“读访问”状态或“无访问”状态的表。类似，用于添加主键或唯一约束的 ALTER TABLE 语句不能引用任何处于“读访问”状态或“无访问”状态的表。
- 不允许 IMPORT 实用程序操作处于“读访问”状态或“无访问”状态的表。
- 不允许 EXPORT 实用程序操作处于“无访问”状态的表，但允许操作处于“读访问”状态的表。如果表处于“读访问”状态，那么 EXPORT 实用程序将只能导出非追加部分中的数据。
- 不允许对处于以下任一状态的表执行涉及在表中移动数据的操作（例如 REORG、REDISTRIBUTE、更新分发密钥、更新多维集群密钥、更新范围集群密钥以及更新表分区密钥等）：读访问、无访问或无数据移动。
- 允许 load、backup、restore、update statistics、runstats、reorgchk、list history 和 rollforward 实用程序对处于以下任一状态的表执行操作：完全访问、读访问、无访问或无数据移动。
- ALTER TABLE、COMMENT、DROP TABLE、CREATE ALIAS、CREATE TRIGGER、CREATE VIEW、GRANT、REVOKE 和 SET INTEGRITY 语句可以引用处于以下任一状态的表：完全访问、读访问、无访问或无数据移动。但是，它们可能会导致将表置于“无访问”状态。
- 在运行时访问处于“无访问”状态的表时，程序包、视图以及任何其他依赖于该表的对象都将返回错误。运行时对表执行插入、更新或删除操作时，依赖于处于“读访问”状态的表的程序包都将返回错误。

通过 SET INTEGRITY 语句除去违例行不是“删除”事件。因此，SET INTEGRITY 语句从不激活触发器。类似，使用 FORCE GENERATED 选项更新生成列不会激活触发器。

- 只要情况允许，就将使用增量处理，这是因为增量处理更有效。大多数情况下，不需要 INCREMENTAL 选项。但是，需要此选项以确保以增量方式处理完整性检查。如果系统检测到需要执行完全处理以确保数据完整性，那么将返回错误（SQLSTATE 55019）。
- 有关使用 IMMEDIATE UNCHECKED 子句的警告：
  - 此子句供实用程序使用，建议不要让应用程序使用此子句。如果表中存在未满足对表定义的完整性规范的数据，且使用了 IMMEDIATE UNCHECKED 选项，那么可能返回错误的查询结果。

    将在目录中记录在未执行必需的完整性处理的情况下使表脱离“设置完整性暂挂”状态这一事实（SYSCAT.TABLES 视图的 CONST_CHECKED 列中的各个字节将设置为“U”）。这指示用户承担了有关特定约束的数据完整性责任。此值保留不变，直到：
    - 将表重新置于“设置完整性暂挂”状态（通过在指定了 OFF 选项的 SET INTEGRITY 语句中引用该表），此时，CONST_CHECKED 列中的“U”值将更改为“W”值，这指示用户先前承担了数据完整性责任，而系统需要验证数据。
    - 将删除表的所有未检查约束。

"W"状态有别于"N"状态，"W"状态记录了一个事实：用户先前检查了完整性，但系统尚未检查完整性。如果用户发出指定了 NOT INCREMENTAL 选项的 SET INTEGRITY ... IMMEDIATE CHECKED 语句，那么系统将重新检查整个表的数据完整性（或者对具体化查询表执行完全刷新），然后将"W"状态更改为"Y"状态。如果指定了 IMMEDIATE UNCHECKED，或者未指定 NOT INCREMENTAL，那么"W"状态将重新更改为"U"以记录一个事实：系统尚未验证某些数据。在后一情况下（即，未指定 NOT INCREMENTAL），将返回警告（SQLSTATE 01636）。

如果已使用 IMMEDIATE UNCHECKED 子句检查了基础表的完整性，那么基础表的 CONST_CHECKED 列中的"U"值将传播至以下表的对应 CONST_CHECKED 列：

- 从属立即具体化查询表
- 从属延迟具体化查询表
- 从属登台表

对于从属立即具体化查询表，每当使基础表脱离"设置完整性暂挂"状态时以及每当刷新具体化查询表时，将完成此传播操作。对于从属延迟具体化查询表，每当刷新具体化查询表时，将完成此传播操作。对于从属登台表，每当使基础表脱离"设置完整性暂挂"状态时，将完成此传播操作。从属具体化查询表和登台表的 CONST_CHECKED 列中的这些传播"U"值记录了一个事实：这些具体化查询表和登台表依赖于某个基础表，而该基础表的必需完整性处理已使用 IMMEDIATE UNCHECKED 选项绕过。

对于具体化查询表，已由基础表传播的 CONST_CHECKED 列中的"U"值将保持不变，直到具体化查询表完全刷新并且它的任何基础表中的对应 CONST_CHECKED 列都没有"U"值。完成此类刷新后，具体化查询表的 CONST_CHECKED 列中的"U"值将更改为"Y"。

对于登台表，由基础表传播的 CONST_CHECKED 列中的"U"值将保持不变，直到刷新了登台表的对应延迟具体化查询表。完成此类刷新后，登台表的 CONST_CHECKED 列中的"U"值将更改为"Y"。

– 如果在指定了 IMMEDIATE CHECKED 选项的同一 SET INTEGRITY 语句中检查了子表及其父表，并且父表要求对其约束进行完全检查，那么子表将检查其外键约束，而不管子表的 CONST_CHECKED 列中的外键约束值是否为"U"。

- 使用 LOAD INSERT 或 ALTER TABLE ATTACH 追加数据后，指定了 IMMEDIATE CHECKED 选项的 SET INTEGRITY 语句将检查表是否违反了约束。系统确定是否可以对表执行增量处理。如果可以，那么将仅检查追加的部分是否存在完整性违例。如果不能，那么系统将检查整个表是否存在完整性违例。
- 考虑语句：

```
SET INTEGRITY FOR T IMMEDIATE CHECKED
```

在以下情况下，系统将需要执行完全刷新或者将检查整个表的完整性（不能指定 INCREMENTAL 选项）：

– 如果已将新约束添加至 T 自身，而 T 处于"设置完整性暂挂"状态

– 如果对 T、其父表或其基础表执行了 LOAD REPLACE 操作

- 如果最近对 T、其父表或其基础表执行完整性检查后，激活了 NOT LOGGED INITIALLY WITH EMPTY TABLE 选项
- 以非增量方式检查 T 的任何父表或 T 的基础表（如果 T 是具体化查询表或登台表）的完整性后对完全处理进行了级联
- 如果已将包含表或其父表（或者具体化查询表或登台表的基础表）的表空间前滚到某一时间点，而表及其父表或基础表（如果表是具体化查询表或登台表）位于不同表空间中
- 如果 T 是具体化查询表，并且最近执行刷新后已直接对 T 执行了 LOAD REPLACE 或 LOAD INSERT 操作

- 如果未满足在上一条目中描述的完全处理条件，那么当用户未对语句 SET INTEGRITY FOR T IMMEDIATE CHECKED 指定 NOT INCREMENTAL 选项时，系统将尝试仅检查已追加部分的完整性，或者执行增量刷新（如果它是具体化查询表）。
- 如果在执行完整性处理期间发生错误，那么将回滚所有处理效果（包括从原始表中删除并插入异常表中）。
- 如果发出指定了 FORCE GENERATED 选项的 SET INTEGRITY 语句由于缺乏日志空间而失败，那么请增加可用的活动日志空间并重新发出 SET INTEGRITY 语句。另外，还可以使用指定了 GENERATED COLUMN 和 IMMEDIATE UNCHECKED 选项的 SET INTEGRITY 语句来绕过对表的生成列检查。然后，发出指定了 IMMEDIATE CHECKED 选项（不指定 FORCE GENERATED 选项）的 SET INTEGRITY 语句来检查表是否存在其他完整性违例（如果适用）并使表脱离“设置完整性暂挂”状态。在表脱离“设置完整性暂挂”状态后，可通过在 UPDATE 语句中将缺省（生成）值赋给关键字 DEFAULT 来将生成列更新为这些缺省（生成）值。可通过使用多条基于范围的搜索式 UPDATE 语句（每条语句后跟有落实语句）或通过使用间歇性落实语句的、基于游标的方法来实现此目的。如果在执行使用基于游标的方法的间歇性落实语句后要保留锁定，那么应该使用“with hold”游标。
- 如果使用 SET INTEGRITY 语句或 LOAD 命令的 CASCADE DEFERRED 选项，或者通过指定了 ATTACH 子句的 ALTER TABLE 语句将表置于“设置完整性暂挂”状态，并且使用 SET INTEGRITY 语句的 IMMEDIATE CHECKED 选项检查该表是否存在完整性违例，那么该表会根据需要将其派生外键表、派生立即具体化查询表以及派生立即登台表置于“设置完整性暂挂”状态:
  - 如果检查整个表是否存在完整性违例，那么会将该表的派生外键表、派生立即具体化查询表和派生立即登台表置于“设置完整性暂挂”状态。
  - 如果以增量方式检查表是否存在完整性违例，那么会将该表的派生立即具体化查询表和登台表置于“设置完整性暂挂”状态，而该表的派生外键表将继续处于其原始状态。
  - 如果表根本不需要检查，那么其派生立即具体化查询表、派生登台表及其派生外键表将继续处于它们的原始状态。
- 如果使用 SET INTEGRITY 语句或 LOAD 命令的 CASCADE DEFERRED 选项将表置于“设置完整性暂挂”状态，并且通过 SET INTEGRITY 语句的 IMMEDIATE UNCHECKED 选项使该表脱离“设置完整性暂挂”状态，那么该表会根据需要将其派生外键表、派生立即具体化查询表和派生立即登台表置于“设置完整性暂挂”状态:
  - 如果已使用 REPLACE 方式装入了表，那么会将该表的派生外键表、派生立即具体化查询表和派生立即登台表置于“设置完整性暂挂”状态。
  - 如果已使用 INSERT 方式装入了表，那么会将该表的派生立即具体化查询表和登台表置于“设置完整性暂挂”状态，而该表的派生外键表将继续处于其原始状态。

– 如果未装入表，那么其派生立即具体化查询表、派生登台表及其派生外键表将继续处于它们的原始状态。

- SET INTEGRITY 通常是长时间运行语句。鉴于此，为了避免由于锁定超时而导致回滚整条语句的风险，可在执行 SET INTEGRITY 语句前发出指定了 WAIT 选项的 SET CURRENT LOCK TIMEOUT 语句，然后在落实事务后，将专用寄存器重置为其先前值。但是，请注意 CURRENT LOCK TIMEOUT 专用寄存器仅影响特定的一组锁定类型。
- 如果您使用 ALLOW QUERY OPTIMIZATION USING REFRESH DEFERRED TABLES WITH REFRESH AGE ANY 选项，请确保维护顺序对于 REFRESH DEFERRED 具体化查询表是正确的。例如，考虑两个具体化查询表（MQT1 和 MQT2），这两个具体化查询共享相同的基础表。可使用 MQT1 取代基础表来计算 MQT2 的具体化查询。如果使用单独的语句来维护这两个具体化查询表，并且首先维护 MQT2，那么系统可能选择使用尚未维护的 MQT1 的内容来维护 MQT2。在这种情况下，MQT1 将包含当前数据，但 MQT2 仍可能包含旧数据，即使对 MQT1 和 MQT2 进行维护的时间几乎相同。如果使用两条 SET INTEGRITY 语句，而不是一条，那么正确的维护顺序是首先维护 MQT1。
- 使用 SET INTEGRITY 语句对已装入或已连接的基本表执行完整性处理时，建议您在同一 SET INTEGRITY 语句中处理该基本表的从属 REFRESH IMMEDIATE 具体化查询表和该基本表的 PROPAGATE IMMEDIATE 登台表，以避免在结束 SET INTEGRITY 处理时将这些从属表置于“设置完整性暂挂无访问”状态。注意，对于具有大量从属 REFRESH IMMEDIATE 具体化查询表和 PROPAGATE IMMEDIATE 登台表的基本表，内存约束可能会导致无法在同一语句中对所有从属表进行处理，这一点与基本表不同。
- 如果指定了 FORCE GENERATED 或 GENERATE IDENTITY 选项，并且生成的列是唯一索引的一部分，那么 SET INTEGRITY 语句将返回错误（SQLSTATE 23505）并在检测到唯一索引中存在重复键时回滚。即使所处理的表存在异常表，也将返回此错误。

  在以下情况下会出现此场景:

  – 在 LOAD 命令之后对表运行 SET INTEGRITY 语句，并且在执行装入操作期间指定了 GENERATEDOVERRIDE 或 IDENTITYOVERRIDE 文件类型修饰符。为了防止此场景，建议使用 GENERATEDIGNORE 或 GENERATEDMISSING 文件类型修饰符取代 GENERATEDOVERRIDE，并建议使用 IDENTITYIGNORE 或 IDENTITYMISSING 修饰符取代 IDENTITYOVERRIDE。使用建议的修饰符将避免在执行 SET INTEGRITY 语句期间处理任何 GENERATED BY EXPRESSION 列或标识列。

  – 在运行对 GENERATED BY EXPRESSION 列的表达式进行更改的 ALTER TABLE 语句后，运行了 SET INTEGRITY 语句。

  要在遇到此类场景后使表脱离“设置完整性暂挂”状态:

  – 不要使用 FORCE GENERATED 或 GENERATE IDENTITY 选项来重新生成列值。相反，请将 IMMEDIATE CHECKED 选项与 FOR EXCEPTION 选项一起使用，来将任何违反生成列表达式的行移入异常表。然后，将行从异常表重新插入表，这将生成正确的表达式并执行唯一键检查。由于只需再次处理那些违反了生成列表达式的行，因此将避免重新处理整个表。

  - 如果所处理的表连接了分区，请拆离那些分区，然后再执行上一条目中描述的操作。随后，重新连接分区并执行 SET INTEGRITY 语句以单独在所连接分区上处理完整性。
- 如果对 SET INTEGRITY 语句同时指定了保护表和异常表，那么必须满足以下所有表条件；否则，将返回错误（SQLSTATE 428A5）：
  - 必须使用同一安全策略保护这些表。
  - 如果保护表中的列具有数据类型 DB2SECURITYLABEL，那么异常表中的对应列也必须具有数据类型 DB2SECURITYLABEL。
  - 如果使用安全标号对保护表中的列进行了保护，那么也必须使用同一安全标号对异常表中的对应列进行保护。
- ***兼容性***
  - 为了兼容先前版本的 DB2:
    - 可指定 SET CONSTRAINTS 以取代 SET INTEGRITY
    - 可指定 SUMMARY 以取代 MATERIALIZED QUERY

示例

示例 1: 如下举例说明了某一查询，该查询可提供有关表的“设置完整性暂挂”状态和“设置与完整性相关的访问限制”状态的信息。SUBSTR 用于抽取 SYSCAT.TABLES 的 CONST_CHECKED 列的各个字节。第一个字节表示外键约束；第二个字节表示检查约束；第五个字节表示具体化查询表完整性；第六个字节表示生成列约束；第七个字节表示登台表完整性；第八个字节表示数据分区约束。STATUS 提供“设置完整性暂挂”状态，而 ACCESS_MODE 提供“设置与完整性相关的访问限制”状态。

```
SELECT TABNAME, STATUS, ACCESS_MODE,
  SUBSTR(CONST_CHECKED,1,1) AS FK_CHECKED,
  SUBSTR(CONST_CHECKED,2,1) AS CC_CHECKED,
  SUBSTR(CONST_CHECKED,5,1) AS MQT_CHECKED,
  SUBSTR(CONST_CHECKED,6,1) AS GC_CHECKED,
  SUBSTR(CONST_CHECKED,7,1) AS STG_CHECKED,
  SUBSTR(CONST_CHECKED,8,1) AS DP_CHECKED
FROM SYSCAT.TABLES
```

示例 2: 将 PARENT 表置于“设置完整性暂挂无访问”状态并立即将“设置完整性暂挂”状态级联至其后代。

```
SET INTEGRITY FOR PARENT OFF
  NO ACCESS CASCADE IMMEDIATE
```

示例 3: 将 PARENT 表置于“设置完整性暂挂读访问”状态，而不立即将“设置完整性暂挂”状态级联至其后代。

```
SET INTEGRITY FOR PARENT OFF
  READ ACCESS CASCADE DEFERRED
```

示例 4: 检查表 FACT_TABLE 的完整性。如果未检测到任何完整性违例，那么使表脱离“设置完整性暂挂”状态。如果检测到任何完整性违例，那么将回滚整条语句，而表将继续处于“设置完整性暂挂”状态。

```
SET INTEGRITY FOR FACT_TABLE IMMEDIATE CHECKED
```

示例 5: 检查 SALES 和 PRODUCTS 表的完整性，然后将违反完整性的行移入异常表 SALES_EXCEPTIONS 和 PRODUCTS_EXCEPTIONS。将使 SALES 和 PRODUCTS 表脱离“设置完整性暂挂”状态，而不管是否存在任何完整性违例。

```
SET INTEGRITY FOR SALES, PRODUCTS IMMEDIATE CHECKED
  FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS,
  IN PRODUCTS USE PRODUCTS_EXCEPTIONS
```

示例 6: 在 MANAGER 表中启用 FOREIGN KEY 约束检查并在 EMPLOYEE 表中启用 CHECK 约束检查，以使用 IMMEDIATE UNCHECKED 选项绕过。

```
SET INTEGRITY FOR MANAGER FOREIGN KEY,
  EMPLOYEE CHECK IMMEDIATE UNCHECKED
```

示例 7: 使用两条 ALTER TABLE 语句向 EMP_ACT 表添加检查约束和外键。指定了 OFF 选项的 SET INTEGRITY 语句用于将表置于“设置完整性暂挂”状态，以便在执行两条 ALTER TABLE 语句时不立即检查约束。在单次通过表期间，指定了 IMMEDIATE CHECKED 选项的单条 SET INTEGRITY 语句用于检查所添加的两个约束。

```
SET INTEGRITY FOR EMP_ACT OFF;
ALTER TABLE EMP_ACT ADD CHECK
  (EMSTDATE <= EMENDATE);
ALTER TABLE EMP_ACT ADD FOREIGN KEY
  (EMPNO) REFERENCES EMPLOYEE;
SET INTEGRITY FOR EMP_ACT IMMEDIATE CHECKED
  FOR EXCEPTION IN EMP_ACT USE EMP_ACT_EXCEPTIONS
```

示例 8: 使用正确的值更新生成列。

```
SET INTEGRITY FOR SALES IMMEDIATE CHECKED
  FORCE GENERATED
```

示例 9: 从不同源通过 LOAD INSERT 追加至 REFRESH IMMEDIATE 具体化查询表（SALES_SUMMARY）的基础表（SALES）。以增量方式检查 SALES 表的数据完整性，并以增量方式刷新 SALES_SUMMARY 表。在此场景下，由于系统选择了增量处理，因此对 SALES 表的完整性检查和对 SALES_SUMMARY 表的刷新均以增量方式执行。在 SALES 表上使用 ALLOW READ ACCESS 选项以允许在对表的已装入部分执行完整性检查时并发读取现有数据。

```
LOAD FROM 2000_DATA.DEL OF DEL
  INSERT INTO SALES ALLOW READ ACCESS;
LOAD FROM 2001_DATA.DEL OF DEL
  INSERT INTO SALES ALLOW READ ACCESS;
SET INTEGRITY FOR SALES ALLOW READ ACCESS IMMEDIATE CHECKED
  FOR EXCEPTION IN SALES USE SALES_EXCEPTIONS;
REFRESH TABLE SALES_SUMMARY;
```

示例 10: 将新分区连接至数据分区表 SALES。以增量方式检查 SALES 表的连接数据中是否存在约束违例，并以增量方式刷新从属 SALES_SUMMARY 表。在这两个表上使用 ALLOW WRITE ACCESS 选项以允许在执行完整性检查时进行并发更新。

```
ALTER TABLE SALES
  ATTACH PARTITION STARTING (100) ENDING (200)
  FROM SOURCE;
SET INTEGRITY FOR SALES ALLOW WRITE ACCESS, SALES_SUMMARY ALLOW WRITE ACCESS
  IMMEDIATE CHECKED FOR EXCEPTION IN SALES
  USE SALES_EXCEPTIONS;
```

示例 11: 从数据分区表 SALES 拆离分区。以增量方式刷新从属 SALES_SUMMARY 表。

```
ALTER TABLE SALES
  DETACH PARTITION 2000_PART INTO ARCHIVE_TABLE;
SET INTEGRITY FOR SALES_SUMMARY
  IMMEDIATE CHECKED;
```

示例 12: 使新的用户维护具体化查询表脱离“设置完整性暂挂”状态。

```
CREATE TABLE YEARLY_SALES
  AS (SELECT YEAR, SUM(SALES)AS SALES
  FROM FACT_TABLE GROUP BY YEAR)
  DATA INITIALLY DEFERRED REFRESH DEFERRED MAINTAINED BY USER

SET INTEGRITY FOR YEARLY_SALES
  ALL IMMEDIATE UNCHECKED
```

LOAD QUERY

在处理期间检查装入操作的状态，并返回表状态。如果未执行装入操作，那么将只返回表状态。要成功调用此命令，还需要连接至同一数据库以及一个单独的 CLP 会话。本地用户或远程用户都可以使用此命令。

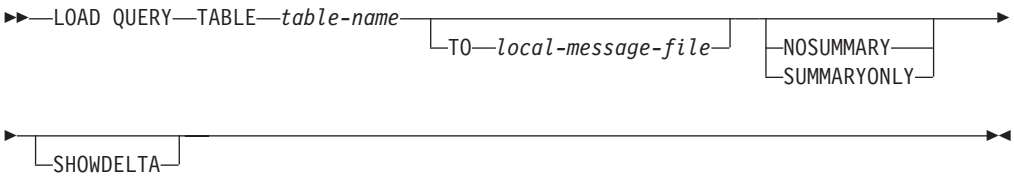
权限

无

必需的连接

数据库

命令语法



命令参数

NOSUMMARY
指定不会报告装入摘要信息（读取的行数、跳过的行数、装入的行数、拒绝的行数、删除的行数、落实的行数以及警告数）。

SHOWDELTA
指定将只报告新信息（与自从上次调用 LOAD QUERY 命令以来所发生的装入事件有关）。

SUMMARYONLY
指定将只报告装入摘要信息。

TABLE *table-name*
指定当前正在将数据装入到的表的名称。如果指定了非限定表名，那么将使用 CURRENT SCHEMA 语句来限定表。

TO *local-message-file*
指定在执行装入操作期间产生的警告和错误消息的目标。此文件不能是为 LOAD 命令指定的 *message-file*。如果该文件已存在，那么会将 LOAD 实用程序所生成的所有消息追加至该文件。

示例

将大量数据装入到 BILLYBOB 数据库的 STAFF 表中的用户想检查装入操作的状态。该用户可指定:

```
db2 connect to billybob
db2 load query table staff to /u/mydir/staff.tempmsg
```

输出文件 /u/mydir/staff.tempmsg 看起来可能类似如下:

```
SQL3501W  由于禁用了数据库正向恢复，因此表所在的表空间将不会被置于"备份暂挂"状态。

SQL3109N  实用程序正开始从文件"/u/mydir/data/staffbig.del"装入数据。

SQL3500W  实用程序正在以下时间开始"装入"阶段:  "2002-03-21
11:31:16.597045"。

   SQL3519W  开始装入一致点。输入记录数 ="0"。     SQL3520W  装入一致点成功。
SQL3519W  开始装入一致点。输入记录数 ="104416"。
   SQL3520W  装入一致点成功。
SQL3519W  开始装入一致点。输入记录数 ="205757"。
   SQL3520W  装入一致点成功。
SQL3519W  开始装入一致点。输入记录数 ="307098"。
   SQL3520W  装入一致点成功。
SQL3519W  开始装入一致点。输入记录数 ="408439"。
   SQL3520W  装入一致点成功。
SQL3532I  LOAD 实用程序当前正处于"装入"阶段。


读取的行数              = 453376
跳过的行数              = 0
装入的行数              = 453376
拒绝的行数              = 0
删除的行数              = 0
落实的行数              = 408439
警告数                  = 0

表状态:
  正在装入
```

使用说明

除了使用锁定以外，LOAD 实用程序还使用表状态来控制对表的访问。可以使用 LOAD QUERY 命令来确定表状态；可以对当前未装入的表使用 LOAD QUERY。对于分区表，报告的状态是相应的可视数据分区状态受到最大限制的结果。例如，如果单个数据分区处于“只读访问”状态，而所有其他数据分区处于“正常”状态，那么装入查询操作将返回“只读访问”状态。装入操作不会让一部分数据分区与表的其余部分处于不同状态。LOAD QUERY 所描述的表状态如下所示:

正常　如果表未处于任何其他（异常）表状态，那么该表处于“正常”状态。“正常”状态是创建表后该表的初始状态。

设置完整性暂挂

该表具有尚未验证的约束。使用 SET INTEGRITY 语句以使该表脱离“设置完整性暂挂”状态。当 LOAD 实用程序开始对带有约束的表执行装入操作时，它就会使该表处于“设置完整性暂挂”状态。

正在装入

这是一种瞬态状态，仅在执行装入操作期间有效。有关装入操作失败或中断时使表脱离“正在装入”状态的信息，请参阅*相关链接*部分中关于装入操作后的暂挂状态的内容。另请参阅“正在装入”表空间状态。

装入暂挂

已经对此表执行了装入操作，但该操作在落实数据前已中止。请发出 LOAD TERMINATE、LOAD RESTART 或 LOAD REPLACE 命令以使表脱离此状态。

只读访问

如果指定了 ALLOW READ ACCESS 选项，那么在执行装入操作期间，表处于此状态。“只读访问”是一种瞬态状态，允许其他应用程序和实用程序对执行装入操作前存在的数据具有读访问权。

REORG 暂挂

已经对表执行了 REORG 命令所建议的 ALTER TABLE 语句。必须执行经典 REORG，才能再次访问表。

不可用 该表不可用。只能删除该表或者从备份复原该表。通过不可恢复的装入操作执行前滚将使表处于不可用状态。

不可重新启动装入

表处于部分装入状态，不允许执行“重新启动装入”操作。该表还将处于装入暂挂状态。发出 LOAD TERMINATE 或 LOAD REPLACE 命令以使该表脱离“不可重新启动装入”状态。如果在装入操作未能成功地重新启动或终止后执行前滚操作，或者根据表处于“正在装入”或“装入暂挂”状态时创建的联机备份执行复原操作，表就会处于“不可重新启动装入”状态。在这两种情况下，“重新启动装入”操作所需的信息均不可靠，“不可重新启动装入”状态导致无法执行“重新启动装入”操作。

未知 LOAD QUERY 命令无法确定表状态。

IBM DB2 数据库产品当前至少支持 25 种表或表空间状态。在特定情况下，这些状态用于控制对数据的访问权，或者根据需要触发特定的用户操作以保护数据库的完整性。其中的大多数状态均由与某个 DB2 实用程序（例如 LOAD 实用程序或者 BACKUP 和 RESTORE 实用程序）的操作相关的事件生成。

虽然在执行装入操作前从属表空间不再处于停顿状态（停顿是一个持久锁），但在执行装入操作期间，“正在装入”表空间状态将阻止对从属表进行备份。“正在装入”表空间状态与“正在装入”表状态的不同之处在于：所有装入操作都使用“正在装入”表状态，但指定了 COPY NO 选项的装入操作（对可恢复数据库执行）还将使用“正在装入”表空间状态。

下表描述了各种受支持的表状态。另外，该表还提供了一些工作示例以说明如何对管理数据库时可能遇到的各种状态作出解释和响应。这些示例均摘自曾在 AIX 上运行的命令脚本；您可以单独复制、粘贴并运行这些示例。如果您是在非 UNIX 系统上运行 DB2 数据库产品，请确保所有路径名均采用适合于您的系统的正确格式。其中的大多数示例均基于随 DB2 数据库产品附带的 SAMPLE 数据库中的表。少数示例需要某些不属于 SAMPLE 数据库的方案，但您可以使用与 SAMPLE 数据库的连接作为起始点。

表 47. 受支持的表状态

状态	示例

装入暂挂	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录），那么创建一个小表空间以包含装入操作的目标表（称为 NEWSTAFF 的新表）：
	`connect to sample;`
	`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256);`
	`create table newstaff like staff in ts1;`
	`load from staffdata.del of del insert into newstaff;`
	`load query table newstaff;`
	`load from staffdata.del of del terminate into newstaff;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY 命令返回的信息显示 NEWSTAFF 表处于“装入暂挂”状态；在执行“装入终止”操作后，表处于“正常”状态。

正在装入	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录）：
	`connect to sample;`
	`create table newstaff like staff;`
	`load from staffdata.del of del insert into newstaff;`
	
	运行装入操作时，请从其他会话执行以下脚本:
	`connect to sample;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY 命令返回的信息显示 NEWSTAFF 表处于“正在装入”状态。

正常	`connect to sample;`
	`create table newstaff like staff;`
	`load query table newstaff;`
	
	LOAD QUERY 命令返回的信息显示 NEWSTAFF 表处于“正常”状态。

表 47. 受支持的表状态 (续)

状态	示例

不可重新启动装入	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录）：
	`update db cfg for sample using logretain recovery;`
	`backup db sample;`
	`connect to sample;`
	`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256);`
	`create table newstaff like staff in ts1;`
	`connect reset;`
	`backup db sample;`
	
	此备份映像的时间戳记为: 20040629205935
	`connect to sample;`
	`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
	`connect reset;`
	`restore db sample taken at 20040629205935;`
	`rollforward db sample to end of logs and stop;`
	`connect to sample;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY 命令返回的信息显示 NEWSTAFF 表处于“不可重新启动装入”和“装入暂挂”状态。
	`connect to sample;`
	`load from staffdata.del of del terminate into newstaff copy yes to /home/melnyk/backups;`
	`load query table newstaff;`
	`connect reset;`
	
	LOAD QUERY 命令返回的信息显示 NEWSTAFF 表当前处于“正常”状态。

只读访问	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录）：
	`connect to sample;`
	`export to st_data.del of del select * from staff;`
	`create table newstaff like staff;`
	`import from st_data.del of del insert into newstaff;`
	`load from staffdata.del of del insert into newstaff allow read access;`
	
	运行装入操作时，请从其他会话执行以下脚本:
	`connect to sample;`
	`load query table newstaff;`
	`select * from newstaff;`
	`connect reset;`
	
	LOAD QUERY 命令返回的信息显示 NEWSTAFF 表处于“只读访问”和“正在装入”状态。该查询仅返回 STAFF 表的已导出内容（在执行装入操作前存在于 NEWSTAFF 中的数据）。

设置完整性暂挂	如果装入输入文件 staff_data.del 包含以下内容:
	
	11,″Melnyk″,20,″Sales″,10,70000,15000:
	`connect to sample;`
	`alter table staff add constraint max_salary check (100000 - salary > 0);`
	`load from staff_data.del of del insert into staff;`
	`load query table staff;`
	
	LOAD QUERY 命令返回的信息显示 STAFF 表处于“设置完整性暂挂”状态。

表 47. 受支持的表状态 (续)

状态	示例

不可用	如果装入输入文件 staff_data.del 包含以下内容:
	
	11,″Melnyk″,20,″Sales″,10,70000,15000:
	
	`update db cfg for sample using logretain recovery;`
	`backup db sample;`
	
	此备份映像的时间戳记为: 20040629182012
	
	`connect to sample;`
	`load from staff_data.del of del insert into staff nonrecoverable;`
	`connect reset;`
	`restore db sample taken at 20040629182012;`
	`rollforward db sample to end of logs and stop;`
	`connect to sample;`
	`load query table staff;`
	`connect reset;`
	
	LOAD QUERY 命令返回的信息显示 STAFF 表处于“不可用”状态。

有关表状态的其他信息，请参阅*相关链接*部分。

还可以使用 LIST UTILITIES 命令来监视装入操作的进度。

LIST TABLESPACES

列示表空间及当前数据库的表空间的信息。

此命令显示的信息也可以在表空间快照中找到。

作用域

此命令仅返回执行此命令的数据库分区的信息。

权限

为下列其中一项:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- LOAD 权限

必需的连接

数据库

命令语法

命令参数

SHOW DETAIL

如果未指定此选项，那么将仅提供有关各个表空间的如下基本信息:

- 表空间标识
- 名称
- 类型（系统管理的空间或数据库管理的空间）
- 内容（任何数据、长整型或索引数据或者临时数据）
- 状态（一个指示当前表空间状态的十六进制值）。表空间的外部可视状态由特定状态值的十六进制之和组成。例如，如果状态为“停顿: EXCLUSIVE”和“装入暂挂”，那么值为 0x0004 + 0x0008，即 0x000c。db2tbst（获取表空间状态）命令可用来获取与给定十六进制值相关联的表空间状态。以下是 sqlutil.h 中列示的位定义:

```
0x0          正常
0x1          停顿: SHARE
0x2          停顿: UPDATE
0x4          停顿: EXCLUSIVE
0x8          装入暂挂
0x10         删除暂挂
0x20         备份暂挂
0x40         正在前滚
0x80         前滚暂挂
0x100        复原暂挂
0x100        恢复暂挂（未使用）
0x200        禁用暂挂
0x400        正在重组
0x800        正在备份
0x1000       必须定义存储器
0x2000       正在复原
0x4000       脱机且不可访问
0x8000       删除暂挂
0x20000      正在装入
0x2000000    可定义存储器
0x4000000    存储器定义处于"最终"状态
0x8000000    前滚前更改了存储器定义
0x10000000   正在进行 DMS 重新平衡
0x20000000   正在删除表空间
0x40000000   正在创建表空间
```

如果指定了此选项，那么将提供有关每个表空间的以下附加信息:

- 总页数
- 可用页数
- 已使用页数
- 空闲页数
- 高水位标记（页数）
- 页大小（字节数）
- 扩展数据块大小（页数）
- 预取大小（页数）
- 容器数
- 最短恢复时间（只有在不为零时才显示）
- 状态更改表空间标识（只有在表空间状态为“装入暂挂”或“删除暂挂”时才显示）

- 状态更改对象标识（只有在表空间状态为“装入暂挂”或“删除暂挂”时才显示）
- 停顿者数目（只有在表空间状态为“停顿: SHARE”、“停顿: UPDATE”或“停顿: EXCLUSIVE”时才显示）
- 每个停顿者的表空间标识和对象标识（只有在停顿者数目大于零时才显示）

示例

以下是 LIST TABLESPACES SHOW DETAIL 的两个样本输出。

```
            当前数据库的表空间
表空间标识                              = 0
名称                                    = SYSCATSPACE
类型                                    = 数据库管理的空间
内容                                    = 任何数据
状态                                    = 0x0000
  详细说明:
    正常
总页数                                  = 895
可用页数                                = 895
已使用页数                              = 895
空闲页数                                = 不适用
高水位标记（页数）                      = 不适用
页大小（字节数）                        = 4096
扩展数据块大小（页数）                  = 32
预取大小（页数）                        = 32
容器数                                  = 1

表空间标识                              = 1
名称                                    = TEMPSPACE1
类型                                    = 系统管理的空间
内容                                    = 临时数据
状态                                    = 0x0000
  详细说明:
    正常
总页数                                  = 1
可用页数                                = 1
已使用页数                              = 1
空闲页数                                = 不适用
高水位标记（页数）                      = 不适用
页大小（字节数）                        = 4096
扩展数据块大小（页数）                  = 32
预取大小（页数）                        = 32
容器数                                  = 1

表空间标识                              = 2
名称                                    = USERSPACE1
类型                                    = 数据库管理的空间
内容                                    = 任何数据
状态                                    = 0x000c
  详细说明:
    停顿: EXCLUSIVE     装入暂挂
总页数                                  = 337
可用页数                                = 337
已使用页数                              = 337
空闲页数                                = 不适用
高水位标记（页数）                      = 不适用
页大小（字节数）                        = 4096
扩展数据块大小（页数）                  = 32
预取大小（页数）                        = 32
容器数                                  = 1
状态更改表空间标识                      = 2
状态更改对象标识                        = 3
```

```
停顿者数目                              = 1
  停顿者 1:
    表空间标识                          = 2
    对象标识                            = 3
DB21011I  在分区数据库服务器环境中，仅列示当前节点上的表空间。

                当前数据库的表空间
表空间标识                              = 0
名称                                    = SYSCATSPACE
类型                                    = 系统管理的空间
内容                                    = 任何数据
状态                                    = 0x0000
  详细说明:
    正常
总页数                                  = 1200
可用页数                                = 1200
已使用页数                              = 1200
空闲页数                                = 不适用
高水位标记（页数）                      = 不适用
页大小（字节数）                        = 4096
扩展数据块大小（页数）                  = 32
预取大小（页数）                        = 32
容器数                                  = 1

表空间标识                              = 1
名称                                    = TEMPSPACE1
类型                                    = 系统管理的空间
内容                                    = 临时数据
状态                                    = 0x0000
  详细说明:
    正常
总页数                                  = 1
可用页数                                = 1
已使用页数                              = 1
空闲页数                                = 不适用
高水位标记（页数）                      = 不适用
页大小（字节数）                        = 4096
扩展数据块大小（页数）                  = 32
预取大小（页数）                        = 32
容器数                                  = 1

表空间标识                              = 2
名称                                    = USERSPACE1
类型                                    = 系统管理的空间
内容                                    = 任何数据
状态                                    = 0x0000
  详细说明:
    正常
总页数                                  = 1
可用页数                                = 1
已使用页数                              = 1
空闲页数                                = 不适用
高水位标记（页数）                      = 不适用
页大小（字节数）                        = 4096
扩展数据块大小（页数）                  = 32
预取大小（页数）                        = 32
容器数                                  = 1

表空间标识                              = 3
名称                                    = DMS8K
类型                                    = 数据库管理的空间
内容                                    = 任何数据
状态                                    = 0x0000
  详细说明:
    正常
总页数                                  = 2000
可用页数                                = 1952
```

```
 已使用页数                              = 96
 空闲页数                                = 1856
 高水位标记（页数）                      = 96
 页大小（字节数）                        = 8192
 扩展数据块大小（页数）                  = 32
 预取大小（页数）                        = 32
 容器数                                  = 2

 表空间标识                              = 4
 名称                                    = TEMP8K
 类型                                    = 系统管理的空间
 内容                                    = 临时数据
 状态                                    = 0x0000
   详细说明:
      正常
 总页数                                  = 1
 可用页数                                = 1
 已使用页数                              = 1
 空闲页数                                = 不适用
 高水位标记（页数）                      = 不适用
 页大小（字节数）                        = 8192
 扩展数据块大小（页数）                  = 32
 预取大小（页数）                        = 32
 容器数                                  = 1
DB21011I  在分区数据库服务器环境中，仅列示当前节点上的表空间。
```

使用说明

在分区数据库环境中，此命令并不返回数据库中的所有表空间。要获取所有表空间的列表，请查询 SYSCAT.TABLESPACES。

在表空间重新平衡期间，可用页数将包括新添加的容器的页数，但在重新平衡完成前，这些新页不会反映在空闲页数中。如果未进行表空间重新平衡，那么已使用页数加上空闲页数等于可使用页数。

IBM DB2 数据库产品当前至少支持 25 种表或表空间状态。在特定情况下，这些状态用于控制对数据的访问权，或者根据需要触发特定的用户操作以保护数据库的完整性。其中的大多数状态均由与某个 DB2 实用程序（例如 LOAD 实用程序或者 BACKUP 和 RESTORE 实用程序）的操作相关的事件生成。

下表描述了各种受支持的表空间状态。另外，该表还提供了一些工作示例以说明如何对管理数据库时可能遇到的各种状态作出解释和响应。这些示例均摘自曾在 AIX 上运行的命令脚本；您可以单独复制、粘贴并运行这些示例。如果您是在非 UNIX 系统上运行 DB2 数据库产品，请确保所有路径名均采用适合于您的系统的正确格式。其中的大多数示例均基于随 DB2 数据库产品附带的 SAMPLE 数据库中的表。少数示例需要某些不属于 SAMPLE 数据库的方案，但您可以使用与 SAMPLE 数据库的连接作为起始点。

表 48. 受支持的表空间状态

状态	十六进制状态值	描述	示例

备份暂挂	0x20	在执行时间点表空间前滚操作后，或者在对可恢复的数据库执行指定了 COPY NO 选项的装入操作后，表空间将处于此状态。必须备份表空间（或者整个数据库），然后才能使用该表空间。如果未备份表空间，那么可以对该表空间中的表执行查询，但不能进行更新。
		注: 另外，在启用数据库以执行前滚恢复操作后，还必须立即备份该数据库。如果 **logretain** 数据库配置参数已设置为 RECOVERY 或者 **userexit** 数据库配置参数已设置为 YES，那么数据库是可恢复的。在备份数据库前，不能激活或连接至此类数据库。完成数据库备份时，**backup_pending** 信息数据库配置参数的值将设置为 NO。
			1. 如果装入输入文件 staff_data.del 包含以下内容:
			11,″Melnyk″,20,″Sales″,10,70000,15000:
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`load from staff_data.del of del messages load.msg insert into staff copy no;`
			`update staff set salary = 69000 where id = 11;`
			2.
			`update db cfg for sample using logretain recovery;`
			`connect to sample;`

正在备份	0x800	这是一种瞬态状态，仅在执行备份操作期间有效。	发出联机 BACKUP DATABASE 命令:
			`backup db sample online;`
			运行备份操作时，请从其他会话执行以下脚本:
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			或
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			返回的 USERSPACE1 信息显示此表空间处于“正在备份”状态。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

正在进行 DMS 重新平衡	0x10000000	这是一种瞬态状态，仅在执行数据重新平衡操作期间有效。如果将新容器添加至一个定义为数据库管理的空间（DMS）的表空间，或者扩展了现有容器，那么可能会对表空间数据进行重新平衡。*重新平衡*是将表空间扩展数据块从一个位置移到另一个位置以尝试保持数据条带化的过程。*扩展数据块*是容器空间的单位（以页计），而条带是*跨表空间的容器集*的扩展数据块层。	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录）：
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`load from staffdata.del of del insert into newstaff nonrecoverable;`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk /NODE0000/SQL00001/ts1c2' 1024);`
			`list tablespaces;`
			`connect reset;`
			
			返回的 TS1 信息显示此表空间处于“正在进行 DMS 重新平衡”状态。

禁用暂挂	0x200	在执行数据库前滚操作期间，表空间可能处于此状态，但在前滚操作结束时，表空间不应该处于此状态。表空间脱机及事务的补偿日志记录未写入所导致的情况将触发此状态。此表空间状态的出现和后续消失对于用户是透明的。	对此表空间状态的说明不在本文档的范围内，因此未给出示例。
删除暂挂	0x8000	在执行数据库重新启动操作期间，如果发现表空间的一个或多个容器出现问题，那么该表空间将处于此状态。如果数据库的前一个会话异常终止（例如，在电源发生故障期间异常终止），那么必须重新启动此数据库。如果表空间处于“删除暂挂”状态，那么表空间将不可用并且只能删除。	对此表空间状态的说明不在本文档的范围内，因此未给出示例。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

正在装入	0x20000	这是一种瞬态状态，仅在对可恢复数据库执行指定了 COPY NO 选项的装入操作期间有效。另请参阅“正在装入”表状态。	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录）：
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create table newstaff like staff;`
			`load from staffdata.del of del insert into newstaff`
			`copy no;`
			`connect reset;`
			
			运行装入操作时，请从其他会话执行以下脚本:
			`connect to sample;`
			`list tablespaces;`
			`connect reset;`
			
			返回的 USERSPACE1 信息显示此表空间处于“正在装入”状态（及“备份暂挂”）状态。

正常	0x0	如果表空间未处于任何其他（异常）表空间状态，那么该表空间处于“正常”状态。“正常”状态是创建表空间后该表空间的初始状态。	`connect to sample;`
			`create tablespace ts1 managed by database using`
			`(file '/home/melnyk/melnyk/NODE0000/SQL00001`
			`/tsc1' 1024);`
			`list tablespaces show detail;`

脱机且不可访问	0x4000	如果表空间的一个或多个容器出现问题，那么该表空间将处于此状态。容器可能意外重命名、移动或损坏。修正问题并且与该表空间相关联的容器可再次访问后，可通过从数据库断开所有应用程序的连接，然后重新连接至该数据库，来除去此异常状态。另外，还可以发出 ALTER TABLESPACE 语句并指定 SWITCH ONLINE 子句，以从表空间中除去“脱机且不可访问”状态，而不必从数据库断开其他应用程序的连接。	`connect to sample;`
			`create tablespace ts1 managed by database using`
			`(file '/home/melnyk/melnyk/NODE0000/SQL00001`
			`/tsc1' 1024);`
			`alter tablespace ts1 add (file '/home/melnyk/melnyk`
			`/NODE0000/SQL00001/tsc2' 1024);`
			`export to st_data.del of del select * from staff;`
			`create table stafftemp like staff in ts1;`
			`import from st_data.del of del insert into stafftemp;`
			`connect reset;`
			
			将表空间容器 tsc1 重命名为 tsc3，然后尝试查询 STAFFTEMP 表:
			`connect to sample;`
			`select * from stafftemp;`
			
			该查询会返回 SQL0290N（不允许进行表空间访问），并且 LIST TABLESPACES 命令会返回 TS1 的状态值 0x4000（脱机且不可访问）。将表空间容器 tsc3 重命名为 tsc1。此时，查询可成功运行。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

停顿互斥	0x4	对表空间停顿函数进行调用的应用程序对表空间具有互斥（读或写）访问权时，该表空间将处于此状态。可通过发出 QUIESCE TABLESPACES FOR TABLE 命令将表空间显式地置于“停顿互斥”状态。	将表空间设置为“停顿互斥”状态前，请确保表空间处于“正常”状态。
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff exclusive;`
			`connect reset;`
			
			从其他会话执行以下脚本:
			`connect to sample;`
			`select * from staff where id=60;`
			`update staff set salary=50000 where id=60;`
			`list tablespaces;`
			`connect reset;`
			
			返回的 USERSPACE1 信息显示此表空间处于“停顿互斥”状态。

停顿共享	0x1	对表空间停顿函数进行调用的应用程序及并发应用程序对表空间都具有读访问权（但没有写访问权）时，该表空间将处于此状态。可通过发出 QUIESCE TABLESPACES FOR TABLE 命令将表空间显式地置于“停顿共享”状态。	将表空间设置为“停顿共享”状态前，请确保表空间处于“正常”状态。
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff share;`
			`connect reset;`
			
			从其他会话执行以下脚本:
			`connect to sample;`
			`select * from staff where id=40;`
			`update staff set salary=50000 where id=40;`
			`list tablespaces;`
			`connect reset;`
			
			返回的 USERSPACE1 信息显示此表空间处于“停顿共享”状态。

停顿更新	0x2	对表空间停顿函数进行调用的应用程序对表空间具有互斥写访问权时，该表空间将处于此状态。可通过发出 QUIESCE TABLESPACES FOR TABLE 命令将表空间显式地置于“停顿更新”状态。	将表空间设置为“停顿更新”状态前，请确保表空间处于“正常”状态。
			`connect to sample;`
			`quiesce tablespaces for table staff reset;`
			`quiesce tablespaces for table staff intent to update;`
			`connect reset;`
			
			从其他会话执行以下脚本:
			`connect to sample;`
			`select * from staff where id=50;`
			`update staff set salary=50000 where id=50;`
			`list tablespaces;`
			`connect reset;`
			
			返回的 USERSPACE1 信息显示此表空间处于“停顿更新”状态。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

正在重组	0x400	这是一种瞬态状态，仅在执行重组操作期间有效。	发出 REORG TABLE 命令:
			`connect to sample;`
			`reorg table staff;`
			`connect reset;`
			
			运行重组操作时，请从其他会话执行以下脚本:
			`connect to sample;`
			
			1.
			`list tablespaces show detail;`
			
			或
			
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			返回的 USERSPACE1 信息显示此表空间处于“正在重组”状态。
			注: 涉及到 SAMPLE 数据库的“重组表”操作可能在很短时间内完成，因此，使用此方法可能很难察看到“重组进度”状态。

复原暂挂	0x100	在“重定向复原”操作的第一部分之后（即，在发出 SET TABLESPACE CONTAINERS 命令之前），数据库的表空间将处于此状态。必须复原表空间（或整个数据库），然后才能使用该表空间。在成功完成复原操作前，不能连接至数据库。在成功完成复原操作时，**restore_pending** 信息数据库配置参数的值将设置为 NO。	在“可定义存储器”状态下完成“重定向复原”操作的第一部分时，所有表空间都将处于“复原暂挂”状态。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

正在复原	0x2000	这是一种瞬态状态，仅在执行复原操作期间有效。	`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`backup db sample tablespace (userspace1);`
			
			此备份映像的时间戳记为:
			
			20040611174124
			
			`restore db sample tablespace (userspace1) online taken at 20040611174124;`
			
			运行复原操作时，请从其他会话执行以下脚本:
			
			`connect to sample;`
			
			1.
			
			`list tablespaces show detail;`
			
			或
			
			2.
			
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			返回的 USERSPACE1 信息显示此表空间处于“正在复原”状态。

前滚暂挂	0x80	对可恢复数据库执行复原操作后，表空间将处于此状态。必须前滚表空间（或整个数据库），然后才能使用该表空间。如果 **logretain** 数据库配置参数已设置为 RECOVERY 或者 **userexit** 数据库配置参数已设置为 YES，那么数据库是可恢复的。在成功完成前滚操作前，不能激活或连接至数据库。在成功完成前滚操作时，**rollfwd_pending** 信息数据库配置参数的值将设置为 NO。	在“正在复原”状态下完成联机表空间复原操作时，表空间 USERSPACE1 将处于“前滚暂挂”状态。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

正在前滚	0x40	这是一种瞬态状态，仅在执行前滚操作期间有效。	如果装入输入文件 staffdata.del 包含大量数据（例如 20000 或更多记录）：
			`update db cfg for sample using logretain recovery;`
			`backup db sample;`
			`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /ts1c1' 1024);`
			`create table newstaff like staff in ts1;`
			`connect reset;`
			`backup db sample tablespace (ts1) online;`
			此备份映像的时间戳记为:
			20040630000715
			`connect to sample;`
			`load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups;`
			`connect reset;`
			`restore db sample tablespace (ts1) online taken at 20040630000715;`
			`rollforward db sample to end of logs and stop tablespace (ts1) online;`
			运行前滚操作时，请从其他会话执行以下脚本:
			`connect to sample;`
			1.
			`list tablespaces show detail;`
			或
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			返回的 TS1 信息显示此表空间处于“正在前滚”状态。

可定义存储器	0x2000000	在“重定向复原”操作的第一部分之后（即，在发出 SET TABLESPACE CONTAINERS 命令之前），数据库的表空间将处于此状态。这允许您根据需要重定义容器。	`backup db sample;`
			假定此备份映像的时间戳记是 20040613204955:
			`restore db sample taken at 20040613204955 redirect;`
			`list tablespaces;`
			LIST TABLESPACES 命令返回的信息显示所有表空间都处于“可定义存储器”和“复原暂挂”状态。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例

必须定义存储器	0x1000	如果在执行“重定向到新数据库复原”操作期间省略了“设置表空间容器”阶段，或者如果在“设置表空间容器”阶段无法获取指定的容器，那么数据库的表空间将处于此状态。在“设置表空间容器”阶段无法获取指定容器的原因可能是指定的路径名无效或磁盘空间不足等。	`backup db sample;`
			
			假定此备份映像的时间戳记是 20040613204955:
			`restore db sample taken at 20040613204955 into mydb redirect;`
			`set tablespace containers for 2 using (path 'ts2c1');`
			`list tablespaces;`
			
			LIST TABLESPACES 命令返回的信息显示表空间 SYSCATSPACE 和表空间 TEMPSPACE1 处于“必须定义存储器”、“可定义存储器”和“复原暂挂”状态。“必须定义存储器”状态优先于“可定义存储器”状态。

正在创建表空间	0x40000000	这是一种瞬态状态，仅在执行“创建表空间”操作期间有效。	`connect to sample;`
			`create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
			`create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc2' 1024);`
			`create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc3' 1024);`
			
			运行“创建表空间”操作时，请从其他会话执行以下脚本:
			`connect to sample;`
			
			1.
			`list tablespaces show detail;`
			
			或
			
			2.
			`get snapshot for tablespaces on sample;`
			`connect reset;`
			
			返回的 TS1、TS2 和 TS3 信息说明这些表空间处于“正在创建表空间”状态。

表 48. 受支持的表空间状态 (续)

状态	十六进制状态值	描述	示例
正在删除表空间	0x20000000	这是一种瞬态状态，仅在执行“删除表空间”操作期间有效。	connect to sample; create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc1' 1024); create tablespace ts2 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc2' 1024); create tablespace ts3 managed by database using (file '/home/melnyk/melnyk/NODE0000/SQL00001 /tsc3' 1024); drop tablespace ts1; drop tablespace ts2; drop tablespace ts3; 运行“删除表空间”操作时，请从其他会话执行以下脚本: connect to sample; 1. list tablespaces show detail; **或** 2. get snapshot for tablespaces on sample; connect reset; 返回的 TS1、TS2 和 TS3 信息说明这些表空间处于“正在删除表空间”状态。

有关表空间状态的其他信息，请参阅*相关链接*部分。

第 5 章 其他数据移动选项

使用 DB2 Connect 移动数据

如果您正在复杂的环境中工作，在该环境中，需要在主机数据库系统与工作站之间移动数据，那么可以使用 DB2 Connect，这是主机与数据库之间的数据传输网关（请参阅图 17）。

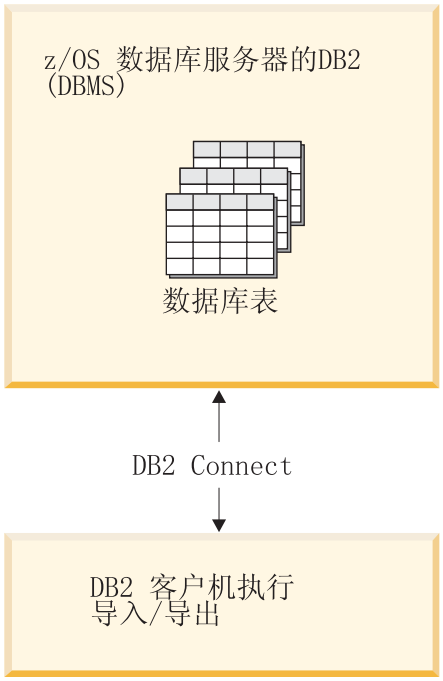


图 17. 通过 DB2 Connect 进行导入/导出

DB2 的 export 和 import 实用程序允许在主机或 System i 服务器数据库与 DB2 Connect 工作站上的文件之间移动数据。然后，可以在任何其他支持这种导出或导入格式的应用程序或关系数据库管理系统中使用该数据。例如，可以将数据从主机或 System i 服务器数据库导出到 PC/IXF 文件中，然后将其导入至 DB2 Windows 版数据库。

可以从数据库客户机或 DB2 Connect 工作站执行导出和导入操作。

注:

1. 要导出或导入的数据必须与适用于这两种数据库的大小和数据类型限制相符。
2. 为了提高导入性能，可以使用复合查询。在 import 实用程序中指定 `compound` 文件类型修饰符以将指定数目的查询语句分组成块。这可以降低网络开销并缩短响应时间。

对于 DB2 Connect 来说，导出和导入操作必须符合下列条件:

- 文件类型必须是 PC/IXF。

- 必须先在目标服务器上创建属性与数据兼容的目标表，然后才能将该数据导入其中。可以使用 db2look 实用程序来获取源表属性。在通过 DB2 Connect 进行导入时，由于只支持 INSERT 选项，所以不能创建表。

如果不符合任何条件，操作会失败并返回错误消息。

注: 导出时不会存储索引定义，导入时也不会使用该定义。

如果导出或导入混合数据（既包含单字节数据也包含双字节数据的列），请考虑下列事项:

- 在使用 EBCDIC 来存储数据的系统（MVS、OS/390、OS/400、VM 和 VSE）上，shift-out 和 shift-in 字符标记双字节数据的开始与结束。在定义数据库表的列长度时，务必为这些字符预留足够的空间。
- 除非列数据有一致的模式，否则建议使用变长字符列。

将数据从工作站移至主机服务器

要将数据移至主机或 System i 服务器数据库:

1. 将 DB2 表中的数据导出至 PC/IXF 文件
2. 使用 INSERT 选项，将 PC/IXF 文件导入到主机服务器数据库中兼容的表中。

要将数据从主机服务器数据库移至工作站:

1. 将主机服务器数据库表中的数据导出到 PC/IXF 文件。
2. 将 PC/IXF 文件导入到 DB2 表中。

示例

以下示例说明如何将数据从工作站移至主机或者 System i 服务器数据库。

通过发出以下命令，将数据导出到 IXF 格式的外部文件中:

```
db2 export to staff.ixf of ixf select * from userid.staff
```

发出以下命令以便对目标 DB2 数据库建立 DRDA 连接:

```
db2 connect to cbc664 user admin using xxx
```

如果目标表不存在，那么在目标 DB2 数据库实例上创建目标表:

```
CREATE TABLE mydb.staff (ID SMALLINT NOT NULL, NAME VARCHAR(9),
             DEPT SMALLINT, JOB CHAR(5), YEARS SMALLINT, SALARY DECIMAL(7,2),
             COMM DECIMAL(7,2))
```

要导入数据，请发出以下命令:

```
db2 import from staff.ixf of ixf insert into mydb.staff
```

这将从 IXF 格式的文件中读取每一数据行，并且将发出 SQL INSERT 语句以将该行插入到 mydb.staff 表中。将持续一行一行地插入数据，直到将所有数据都移入目标表为止。

IBM 红皮书™出版物“Moving Data Across the DB2 Family”中提供了详细信息。可以在以下 URL 位置中找到此红皮书出版物: http://www.redbooks.ibm.com/redbooks/SG246905。

IBM 复制工具（按组件）

IBM 提供了两种基本复制解决方案: Q 复制和 SQL 复制。

Q 复制的主要组件为 Q Capture 程序和 Q Apply 程序。SQL 复制的主要组件为 Capture 程序和 Apply 程序。这两种类型的复制都共享“复制报警监视器”工具。可以使用复制中心和 ASNCLP 命令行程序来设置并管理这些复制组件。

以下列表简要总结了这些复制组件:

Q Capture 程序

阅读 DB2 恢复日志并查找对 DB2 源表所作的更改，然后将已落实的源数据转换为可以 XML 格式发布至预订应用程序的 WebSphere® MQ 消息，或者转换为可以压缩格式复制至 Q Apply 程序的 WebSphere MQ 消息。

Q Apply 程序

从队列中获取 WebSphere MQ 消息、将这些消息变换为 SQL 语句并更新目标表或存储过程。受支持的目标包括 DB2 数据库或子系统，以及通过联合服务器昵称访问的 Oracle、Sybase、Informix® 和 Microsoft® SQL Server 数据库。

Capture 程序

阅读 DB2 恢复日志以获取对已注册的源表或视图所作的更改，然后将已落实的事务数据登台到称为更改数据（CD）表的关系表中，这些数据将存储在 CD 表中直到目标系统准备复制它们为止。SQL 复制还提供了 Capture 触发器，这些触发器使用对非 DB2 数据源表的更改记录填充称为一致更改数据（CCD）表的登台表。

Apply 程序

从登台表中读取数据并对目标进行适当更改。对于非 DB2 数据源，Apply 程序通过联合数据库中的昵称读取 CCD 表并对目标表进行适当更改。

复制报警监视器

用于检查 Q Capture、Q Apply、Capture 和 Apply 程序的运行状况的实用程序。它检查程序终止、发出警告或错误消息、到达指定值的阈值或者执行某个特定操作这些情况，然后将通知发出至电子邮件服务器、寻呼机或 z/OS 控制台。

使用复制中心可以:

- 定义注册、预订、发布、队列映射、报警条件和其他对象。
- 启动、停止、暂挂、恢复和重新初始化复制程序。
- 指定自动复制的时间。
- 指定对数据的 SQL 增强功能。
- 定义源表与目标表之间的关系。

复制模式

db2move 实用程序和 ADMIN_COPY_SCHEMA 过程允许您快速生成数据库模式的副本。在建立模型模式后，可以将它用作创建新版本的模板。

使用 ADMIN_COPY_SCHEMA 过程在相同数据库内复制单个模式，或将 db2move 实用程序与 -co COPY 操作配合使用以将单个模式或多个模式从源数据库复制至目标数据库。源模式中的大多数数据库对象被复制至新模式下的目标数据库。

故障诊断提示

ADMIN_COPY_SCHEMA 过程和 db2move 实用程序都调用 LOAD 命令。在处理装入时，将使数据库目标对象所在的表空间处于“备份暂挂”状态。

ADMIN_COPY_SCHEMA 过程

通过使用指定了 COPYNO 选项的此过程，使目标对象所在的表空间处于“备份暂挂”状态，如上面的说明中所述。要使表空间脱离“设置完整性暂挂”状态，此过程可发出 SET INTEGRITY 语句。在目标表对象定义了引用约束的情况下，也会使目标表处于“设置完整性暂挂”状态。因为该表已经处于“备份暂挂”状态，所以 ADMIN_COPY_SCHEMA 过程尝试发出 SET INTEGRITY 语句的操作将失败。

要解决这种情况，发出 BACKUP DATABASE 命令以使受影响的表空间脱离“备份暂挂”状态。接着，查看此过程生成的错误表的 **Statement_text** 列，以查找处于“设置完整性暂挂”状态的表的列表。然后，对列示的每个表都发出 SET INTEGRITY 语句，以使每个表都脱离“设置完整性暂挂”状态。

db2move 实用程序

此实用程序尝试复制下列类型以外的所有允许的模式对象:

- 表层次结构
- 登台表（多分区数据库环境中的 load 实用程序不支持该表）
- JAR（Java™ 例程归档）
- 昵称
- 程序包
- 视图层次结构
- 对象特权（所有新对象是使用缺省权限创建的）
- 统计信息（新对象不包含统计信息）
- 索引扩展（与用户定义的结构化类型相关）
- 用户定义的结构化类型及其变换函数

不受支持的类型错误

如果在源模式中检测到一个对象的类型是不受支持的类型之一，会将一个条目记录到错误文件中，表明检测到不受支持的对象类型。COPY 操作仍将成功 – 记录此条目是为了通知您此操作未复制这些对象。

未与模式组合的对象

在复制模式操作期间，不会对未与模式组合的对象（例如，表空间和事件监视器）进行操作。应该先在目标数据库上创建这些对象，然后再调用复制模式操作。

已复制的表

复制已复制的表时，不会对复制启用该表的新副本。将该表重新创建为常规表。

不同的实例

如果源数据库与目标数据库不在同一实例中，那么必须对源数据库进行编目。

SCHEMA_MAP 选项

使用 SCHEMA_MAP 选项来指定目标数据库上的其他模式名称时，复制模式操作将仅对对象定义语句执行最小程度的解析，以便将原始模式名称替换为新模式名称。例如，在 SQL 过程的内容中出现的原始模式的任何实例均不会替换为新模式名称。因此，复制模式操作可能无法重新创建这些对象。完成复制操作后，可使用错误文件中的 DDL 来手动重新创建这些失败的对象。

对象之间的相互依赖性

复制模式操作尝试以满足这些对象之间的相互依赖性的顺序来重新创建对象。例如，如果表 T1 包含的某一列引用了用户定义的函数 U1，那么将先重新创建 U1，然后再重新创建 T1。但是，目录中并不随时提供过程的依赖性信息，因此重新创建过程时，复制模式操作将先尝试重新创建所有过程，然后再次尝试重新创建失败的过程（假定失败的过程依赖于上一次尝试期间成功创建的过程，于是在后续尝试期间，将成功地重新创建那些失败的过程）。在后续尝试期间，只要该操作能够成功地重新创建一个或多个失败的过程，就将继续尝试重新创建这些失败的过程。每次尝试重新创建过程时，将在错误文件中记录一个错误（和 DDL）。在错误文件中可能会看到相同过程的多个条目，即使在后续尝试期间，这些过程可能已成功创建。在完成复制模式操作后，您应该查询 SYSCAT.PROCEDURES 表以确定错误文件中列示的这些过程是否已成功地重新创建。

有关更多信息，请参阅 ADMIN_COPY_SCHEMA 过程和 db2move 实用程序。

使用 db2move 实用程序的模式复制的示例

将 db2move 实用程序与 -co COPY 操作配合使用，以便将源数据库中的一个或多个模式复制到目标数据库。在建立模型模式后，可以将它用作创建新版本的模板。

示例 1：使用 -c COPY 选项

以下是 db2move -co COPY 选项的一个示例，它将模式 BAR 从样本数据库复制到目标数据库并将它重命名为 FOO:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u userid -p password
```

创建的新（目标）模式对象与源模式中的对象的对象名相同，但前者具有目标模式限定符。可以创建带有或不带有源表中的数据的表副本。源数据库和目标数据库可以在不同系统上。

示例 2：在 COPY 操作期间指定表空间名称映射

以下示例说明如何在 db2move COPY 操作期间指定要使用的特定表空间名称映射，而不是源系统中的表空间。可指定 SYS_ANY 关键字来指示应使用缺省表空间选择算法来选择目标表空间。在此示例中，db2move 实用程序选择要用作目标的任何可用表空间:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u userid -p password
```

SYS_ANY 关键字可用于所有表空间，或者可以对一些表空间指定特定映射，并对其余表空间指定缺省表空间选择算法:

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
  ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
  -u userid -p password
```

这指示表空间 TS1 映射至 TS2 以及 TS3 映射至 TS4，但其余表空间使用缺省表空间选择算法。

示例 3: 在 COPY 操作后更改对象所有者

在成功执行 COPY 操作后，可以更改在目标模式中创建的每个新对象的所有者。目标对象的缺省所有者是连接用户。如果指定了此选项，那么所有权将转移给新的所有者，如下所示:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
  -u userid -p password
```

目标对象的新所有者是 jrichards。

如果源模式和目标模式位于不同的系统上，那么必须在目标系统上调用 db2move 实用程序。要将模式从一个数据库复制至另一个数据库，此操作需要将从源数据库复制的模式名称列表（用逗号分隔）和目标数据库名。

要复制模式，请从 OS 命令提示符中发出 db2move，如下所示:

```
db2move <dbname> COPY -co <COPY- options>
 -u <userid> -p <password>
```

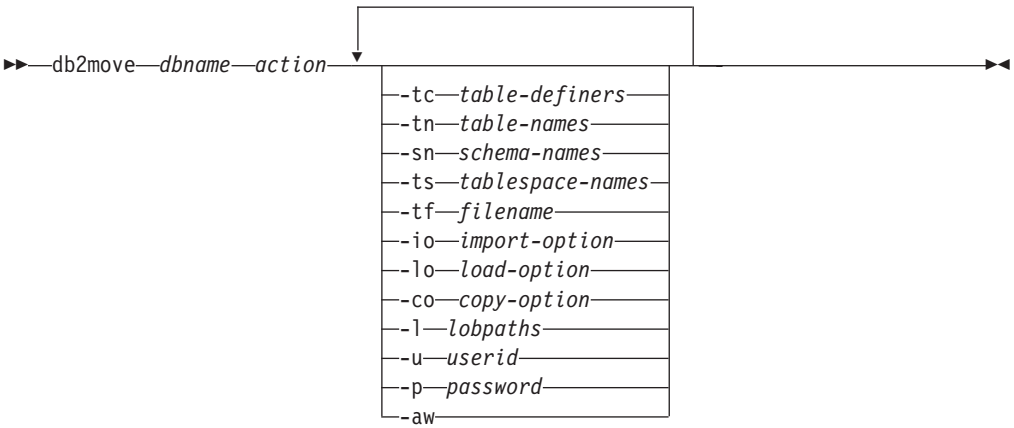
db2move - 数据库移动工具

以 EXPORT/IMPORT/LOAD 方式使用此工具时，将便于在工作站上的 DB2 数据库之间移动大量的表。该工具将查询特定数据库的系统目录表并编译所有用户表的列表。然后以 PC/IXF 格式导出这些表。可以将这些 PC/IXF 文件导入或装入到同一系统上的另一个本地 DB2 数据库；也可以将它们传输至另一个工作站平台，然后导入或装入到该平台上的 DB2 数据库中。当使用此工具时，不会移动具有结构化类型列的表。当以 COPY 方式使用此工具时，它将便于复制模式。

权限

此工具将根据用户请求的操作来调用 DB2 导出、导入和装入 API。因此，发出请求的用户标识必须具有这些 API 所要求的正确权限，否则请求将失败。

命令语法



命令参数

dbname

数据库的名称。

action 必须为下列其中一项:

EXPORT

导出所有满足选项中的过滤条件的表。如果未指定任何选项，那么将导出所有表。内部登台信息存储在 db2move.lst 文件中。

IMPORT

导入内部登台文件 db2move.lst 中所列示的所有表。对于特定于 IMPORT 的操作，请使用 -io 选项。

LOAD 装入内部登台文件 db2move.lst 中所列示的所有表。对于特定于 LOAD 的操作，请使用 -lo 选项。

COPY 将模式复制到目标数据库中。使用 -sn 选项来指定一种或多种模式。有关特定于 COPY 的选项，请参阅 -co 选项。使用 -tn 或 -tf 选项来以 LOAD_ONLY 方式过滤表。

请参阅以下内容以获取在执行每项操作期间生成的文件的列表。

-tc *table-definers*

缺省值是所有定义程序。

这只是一项 EXPORT 操作。如果指定了此选项，那么只会导出由此选项列示的定义程序所创建的那些表。如果未指定此选项，那么缺省情况是使用所有定义程序。当指定多个定义程序时，必须将它们用逗号分隔开；定义程序标识之间不允许存在空格。可以将此选项与 -tn *table-names* 选项一起用来选择要导出的表。

可以使用星号（*）作为通配符，并且可以将星号放在字符串中的任何位置。

-tn *table-names*

缺省值是所有用户表。

这只是一项 EXPORT 或 COPY 操作。

如果与 EXPORT 操作一起指定，那么仅导出表名与指定字符串中的表名相匹配的那些表。如果未指定此选项，那么缺省情况是使用所有用户表。当指定多个表名时，必须将它们用逗号分隔开；表名之间不允许存在空格。表名应列示为“未限定”且 -sn 选项应用于过滤模式。

对于导出操作，可以使用星号（*）作为通配符，并且可以将星号放在字符串中的任何位置。

如果与 COPY 操作一起指定，那么还必须指定 -co ″MODE″ LOAD_ONLY *copy-option*，并且将仅在目标数据库上重新填充那些指定的表。列示表名时应该让它们的模式限定符采用 ″schema″.″table″ 格式。

-sn *schema-names*

适用于 EXPORT 的缺省值是所有模式（但不适用于 COPY）。

如果指定了此选项，那么将导出或复制其模式名称完全匹配的那些表。如果指定了多个模式名称，那么必须将它们用逗号分隔开；模式名称之间不允许存在空格。对于少于 8 个字符的模式名称，会将它填充为 8 个字符。

就导出来说：如果在模式名称中使用了星号（*）作为通配符，那么会将星号更改为百分号（%），并且将在 WHERE 子句的 LIKE 谓词中使用（带有百分号）表名。如果未指定此选项，那么缺省情况是使用所有模式。如果将此选项与 -tn 或 -tc 选项一起使用，那么 db2move 将只对其模式与指定模式名称相匹配、并且其定义程序与指定的定义程序相匹配的那些表执行操作。使用星号作为通配符时，必须将模式名称 `fred` 指定为 `-sn fr*d*` 而不是指定为 `-sn fr*d`。

-ts *tablespace-names*

缺省值是所有表空间。

这只是一项 EXPORT 操作。如果指定了此选项，那么将只导出位于指定表空间中的那些表。如果在表空间名中使用了星号（*）作为通配符，那么会将星号更改为百分号（%），并且将在 WHERE 子句的 LIKE 谓词中使用（带有百分号）表名。如果未指定 -ts 选项，那么缺省情况是使用所有表空间。如果指定了多个表空间名，那么必须将它们用逗号分隔开；表空间名之间不允许存在空格。对于少于 8 个字符的表空间名，会将它填充为 8 个字符。例如，当使用星号作为通配符时，必须将表空间名 `mytb` 指定为 `-ts my*b*` 而不是 `-sn my*b`。

-tf *filename*

如果与 EXPORT 操作一起指定，那么仅导出表名与指定文件中的表名完全匹配的那些表。如果未指定此选项，那么缺省情况是使用所有用户表。每行应列示一个表，并且每个表都应是标准表。不允许在字符串中使用通配符。以下是一个文件的内容示例:

```
"SCHEMA1"."TABLE NAME1"
"SCHEMA NAME77"."TABLE155"
```

如果与 COPY 操作一起指定，那么还必须指定 -co ″MODE″ LOAD_ONLY *copy-option*，并且将仅在目标数据库上重新填充那些在文件中指定的表。列示表名时应该让它们的模式限定符采用 ″schema″.″table″ 格式。

-io *import-option*

缺省值为 REPLACE_CREATE。请参阅“不推荐使用 IMPORT 命令选项 CREATE 和 REPLACE_CREATE”以了解 IMPROT CREATE 函数的局限性。

有效选项包括：INSERT、INSERT_UPDATE、REPLACE、CREATE 和 REPLACE_CREATE。

-lo *load-option*

缺省值是 INSERT。

有效选项为：INSERT 和 REPLACE。

-co 当 db2move 操作为 COPY 时，可紧随 -co 之后指定下列选项:

“TARGET_DB *db name* **[USER** *userid* **USING** *password***]”**

允许用户指定目标数据库的名称以及用户/密码。（可以使用现有的 -p 和 -u 选项来指定源数据库用户/密码）。`USER/USING` 子句是可选的。如果 USER 指定了用户标识，那么必须在 USING 子句后面提供密码。如果未指定密码，那么 db2move 将提示输入密码信息。之所以提示您输入密码信息，是因为下面所讨论的安全性原因。对于 COPY 操作，TARGET_DB 是一个必须选择的选项。TARGET_DB 不能与源数据库

相同。ADMIN_COPY_SCHEMA 过程可以用于复制同一数据库中的模式。COPY 操作要求至少输入一种模式（-sn）或一个表（-tn 或 -tf）。

运行多个 db2move 命令以将模式从一个数据库复制到另一个数据库时将导致死锁。一次只能发出一个 db2move 命令。在复制处理期间对源模式中的表进行更改，可能意味着在复制后目标模式中的数据与之不相同。

“MODE”

DDL_AND_LOAD
根据源模式创建所有受支持的对象，并使用源表数据来填充表。这是缺省选项。

DDL_ONLY
根据源模式创建所有受支持的对象，但是不重新填充表。

LOAD_ONLY
将指定的所有表从源数据库装入到目标数据库中。表必须已存在于目标上。LOAD_ONLY 方式要求使用 -tn 或 -tf 选项来输入一个或多个表。

这是一个只能用于 COPY 操作的可选选项。

“SCHEMA_MAP”

当复制到目标时，允许用户重命名模式。提供源模式与目标模式之间的映射的列表，将这两种模式用逗号分隔，然后用括号将它们括起来。例如，schema_map ((s1, t1), (s2, t2))。这意味着会将模式 s1 中的对象复制到目标上的模式 t1；而将模式 s2 中的对象复制到目标上的模式 t2。缺省情况是目标模式名称与源模式名称相同，并且也建议您这样做。之所以建议您这样做，是因为 db2move 不会尝试修改对象主体中任何限定对象的模式。因此，如果对象主体中有限定对象，那么使用不同的目标模式名称可能会导致问题。

例如：`create view FOO.v1 as 'select c1 from FOO.t1'`

在此例中，将 FOO 模式复制到 BAR 时会重新生成 v1：`create view BAR.v1 as 'select c1 from FOO.t1'`

这将由于目标数据库上不存在 FOO 模式而失败，或者由于 FOO 不同于 BAR 而产生意外的结果。保持与源数据库具有相同模式名称就可以避免这些问题。如果模式之间存在交叉依赖性，那么必须复制所有互相依赖的模式，否则在复制具有交叉依赖性的对象时可能会出错。

例如：`create view FOO.v1 as 'select c1 from BAR.t1'`

在此例中，如果没有同时复制 BAR，那么复制 v1 时将失败；如果目标上的 BAR 不同于源中的 BAR，那么复制 v1 时将产生意外的结果。db2move 将不会尝试检测交叉模式依赖性。

这是一个只能用于 COPY 操作的可选选项。

“NONRECOVERABLE”

此选项允许用户覆盖要使用 COPY-NO 来完成的装入操作的缺省行为。在缺省行为下，将强制用户为要装入到的每个表空间生成备份。当指定此 NONRECOVERABLE 关键字时，将不会强制用户立即生成表空间

的备份。但是，强烈建议您尽快生成备份，以确保可正确地恢复新创建的表。这是一个用于 COPY 操作的可选选项。

"OWNER"

在成功执行 COPY 操作后，允许用户更改在目标模式中创建的每个新对象的所有者。目标对象的缺省所有者将是连接用户；如果指定了此选项，那么会将所有权转移给新的所有者。这是一个用于 COPY 操作的可选选项。

"TABLESPACE_MAP"

在复制期间，用户可指定要使用的表空间名称映射，而不是源系统中的表空间。这将是一些表空间映射的数组，并用括号括起来。例如，`tablespace_map ((TS1, TS2),(TS3, TS4))`。这意味着会将表空间 TS1 中的所有对象复制到目标数据库上的表空间 TS2，而将表空间 TS3 中的对象复制到目标上的表空间 TS4。对于 `((T1, T2),(T2, T3))`，将在目标数据库上的 T2 中重新创建在源数据库上的 T1 中找到的所有对象，而在目标数据库上的 T3 中重新创建在源数据库上的 T2 中找到的所有对象。缺省情况是使用与源数据库中的表空间名相同的表空间名。在这种情况下，不需要此表空间的输入映射。如果指定的表空间不存在，那么复制使用该表空间的对象将失败，并且会记录到错误文件中。

用户还可以选择使用 `SYS_ANY` 关键字来指示应使用缺省表空间选择算法来选择目标表空间。在这种情况下，db2move 将能够选择任何可用表空间来用作目标。`SYS_ANY` 关键字可用于所有表空间。示例：`tablespace_map SYS_ANY`。另外，用户可以对一些表空间指定特定映射，并对其余表空间指定缺省表空间选择算法。例如：`tablespace_map ((TS1, TS2),(TS3, TS4), SYS_ANY)`。这指示表空间 TS1 映射至 TS2 以及 TS3 映射至 TS4，但其余表空间将使用缺省表空间目标。由于不可能有一个以"SYS"开头的表空间，因此将使用 `SYS_ANY` 关键字。

这是一个用于 COPY 操作的可选选项。

-l *lobpaths*

对于 IMPORT 和 EXPORT，如果指定了此选项，那么它还将用于 XML 路径。缺省值是当前目录。

此选项指定将（作为 EXPORT 的一部分）创建 LOB 或 XML 文件或者（作为 IMPORT 或 LOAD 的一部分）搜索 LOB 或 XML 文件的绝对路径名。当指定多个路径时，必须将它们用逗号分隔开；路径之间不允许存在空格。如果指定了多个路径，那么 EXPORT 将以循环方式使用这些路径。它会将一个 LOB 文档写入第一个路径，将下一个 LOB 文档写入第二个路径，以此类推，直到写入最后一个路径，然后又开始写入第一个路径。对于 XML 文档，情况亦如此。（在 IMPORT 或 LOAD 期间）如果在第一个路径中找不到文件，那么将使用第二个路径，以此类推。

-u *userid*

缺省值是已登录的用户标识。

用户标识和密码均为可选。但是，如果指定了其中任何一项，就必须指定另一项。如果在与远程服务器相连的客户机上运行命令，那么应指定用户标识和密码。

-p *password*

缺省值是已登录的密码。用户标识和密码均为可选。但是，如果指定了其中任何一项，就必须指定另一项。当指定了 -p 选项但是未提供密码时，db2move 将提示您输入密码。这样做是为了安全起见。通过命令行输入密码会产生安全性问题。例如，ps -ef 命令就会将密码显示出来。但是，如果通过脚本调用了 db2move，那么将必须提供密码。如果在与远程服务器相连的客户机上发出命令，那么应指定用户标识和密码。

-aw 允许警告。如果未指定 -aw，那么在导出期间产生警告的表将不会包括在 db2move.lst 文件中（尽管仍然会生成表的 .ixf 文件和 .msg 文件）。在某些情况下（例如，数据截断），用户可能想允许将这些表包括在 db2move.lst 文件中。如果指定此选项，那么会允许在导出期间将接收到警告的那些表包括在 .lst 文件中。

示例

- 要导出 SAMPLE 数据库中的所有表（对所有选项都使用缺省值），请发出以下命令:

```
db2move sample export
```

- 要导出由 userid1 或用户标识 LIKE us%rid2 创建的、并且名称为 tbname1 或者表名 LIKE %tbname2 的所有表，请发出以下命令:

```
db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2
```

- 要导入 SAMPLE 数据库中的所有表（将在 LOB 路径 D:\LOBPATH1 和 C:\LOBPATH2 中搜索 LOB 文件；此示例仅适用于 Windows 操作系统），请发出以下命令:

```
db2move sample import -l D:\LOBPATH1,C:\LOBPATH2
```

- 要装入 SAMPLE 数据库中的所有表（将在 /home/userid/lobpath 子目录和 tmp 子目录中搜索 LOB 文件；此示例仅适用于 Linux 和 UNIX 系统），请发出以下命令:

```
db2move sample load -l /home/userid/lobpath,/tmp
```

- 要使用指定的用户标识和密码以 REPLACE 方式导入 SAMPLE 数据库中的所有表，请发出以下命令:

```
db2move sample import -io replace -u userid -p password
```

- 要将 schema1 模式从源数据库 dbsrc 复制到目标数据库 dbtgt 中，请发出以下命令:

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
```

- 要将 schema1 模式从源数据库 dbsrc 复制到目标数据库 dbtgt 中，在目标数据库上将该模式重命名为 newschema1，并将源表空间 ts1 映射至目标数据库上的 ts2，请发出以下命令:

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
     SCHEMA_MAP ((schema1,newschema1)) TABLESPACE_MAP ((ts1,ts2), SYS_ANY))
```

使用说明

- 后跟 db2move IMPORT/LOAD 的 db2move EXPORT 可促进表数据移动。必须手动移动与表相关联的所有其他数据库对象（例如别名、视图或触发器）以及这些表可能依赖于的对象（例如用户定义的类型或用户定义的函数）。
- 如果使用带有 CREATE 或 REPLACE_CREATE 选项的 IMPORT 操作在目标数据库中创建表（这两个选项都不推荐使用，将来的发行版中可能会将它们除去），那么将受到“重新创建导入的表”中所列局限性的限制。在使用指定了 REPLACE_CREATE 选项的 db2move 命令进行导入期间，如果遇到意外错误，请检查相应的 tabnnn.msg 消息文件并考虑这些错误是否由于表创建局限性而导致。

- 使用 db2move 时无法导入或装入包含 GENERATED ALWAYS 标识列的表。但是，您可以手动导入或装入这些表。有关更多信息，请参阅『标识列装入注意事项』或『标识列导入注意事项』。
- 当 db2move 调用导出、导入或装入 API 时，**FileTypeMod** 参数设置为 lobsinfile。即，对于每个表，LOB 数据保存在 PC/IXF 文件之外的文件中。
- 必须在数据库和数据文件所在的机器上以本地方式运行 LOAD 命令。
- 使用 db2move LOAD 且对数据库（该数据库可恢复）启用了 logretain 时:
  - 如果未指定 NONRECOVERABLE 选项，那么将使用缺省 COPY NO 选项 db2move 调用 db2Load API，并且在该实用程序完成时，会将装入表所在的表空间置于“备份暂挂”状态（必须执行数据库或表空间完全备份，才能使表空间脱离“备份暂挂”状态）。
  - 如果指定了 NONRECOVERABLE 选项，那么不会将表空间置于“备份暂挂”状态，但是，如果稍后执行了前滚恢复，那么表将标记为不可访问且必须删除。有关“装入可恢复性”选项的更多信息，请参阅『用于提高装入性能的选项』。
- 可通过改变缺省缓冲池 IBMDEFAULTBP 及通过更新配置参数 **sortheap**、**util_heap_sz**、**logfilsiz** 和 **logprimary** 来提高指定了 IMPORT 或 LOAD 操作的 db2move 命令的性能。

使用 EXPORT 时需要的/生成的文件:

- 输入: 无。
- 输出:

  EXPORT.out
  EXPORT 操作的汇总结果。

  db2move.lst
  原始表名、它们的相应 PC/IXF 文件名（tabnnn.ixf）和消息文件名（tabnnn.msg）的列表。此列表、已导出的 PC/IXF 文件以及 LOB 文件（tabnnnc.yyy）被用作 db2move IMPORT 或 LOAD 操作的输入。

  tabnnn.ixf
  特定表的已导出 PC/IXF 文件。

  tabnnn.msg
  相应表的导出消息文件。

  tabnnnc.yyy
  特定表的已导出 LOB 文件。

  “nnn”是表的编号。“c”是字母表中的一个字母。“yyy”是 001 到 999 范围内的一个数字。

  仅当要导出的表包含 LOB 数据时才会创建这些文件。如果创建了这些 LOB 文件，那么会将它们放入“lobpath”目录中。LOB 文件总共可以有 26,000 个名称。

  system.msg
  这是一个消息文件，它包含创建或删除文件或目录命令产生的系统消息。仅当执行 EXPORT 操作并且指定了 LOB 路径时，才使用此消息文件。

使用 IMPORT 时需要的/生成的文件:

- 输入:

db2move.lst
EXPORT 操作产生的输出文件。

tabnnn.ixf
EXPORT 操作产生的输出文件。

tabnnnc.yyy
EXPORT 操作产生的输出文件。

- 输出:

IMPORT.out
IMPORT 操作的汇总结果。

tabnnn.msg
相应表的导入消息文件。

使用 LOAD 时需要的/生成的文件:

- 输入:

db2move.lst
EXPORT 操作产生的输出文件。

tabnnn.ixf
EXPORT 操作产生的输出文件。

tabnnnc.yyy
EXPORT 操作产生的输出文件。

- 输出:

LOAD.out
LOAD 操作的汇总结果。

tabnnn.msg
相应表的 LOAD 消息文件。

使用 COPY 时需要的/生成的文件:

- 输入: 无
- 输出:

COPYSCHEMA.msg
输出文件，包含执行 COPY 操作期间生成的消息。

COPYSCHEMA.err
输出文件，包含执行 COPY 操作期间遇到的每个错误的错误消息，还包含无法在目标数据库上为每个对象重新创建的 DDL 语句。

LOADTABLE.msg
输出文件，包含 LOAD 实用程序（用于在目标数据库上重新填充数据）的每次调用所生成的消息。

LOADTABLE.err
输出文件，包含装入期间遇到失败的表名或仍需要在目标数据库上进行填充的表的名称。请参阅『重新启动失败的复制模式操作』主题，以了解更多详细信息。

这些文件都具有时间戳记，并且运行一次命令所生成的所有文件都具有相同的时间戳记。

使用自动生成的脚本执行重定向复原

执行重定向复原操作时，需要指定存储在备份映像中的物理容器的位置，并且需要提供每个将要改变的表空间的全部容器。使用以下过程来根据现有备份映像生成重定向复原脚本，修改生成的脚本，然后运行该脚本以执行重定向复原。

仅当先前已使用 DB2 backup 实用程序备份了数据库时，才能执行重定向复原。

- 如果数据库存在，您必须能够连接至该数据库才能生成脚本。因此，如果数据库需要迁移或崩溃恢复，那么必须在尝试生成重定向复原脚本之前进行这些操作。
- 如果正在分区数据库环境中工作，并且目标数据库不存在，那么不能运行该命令来在所有数据库分区上同时生成重定向复原脚本。相反，一次只能在一个数据库分区上运行用于生成重定向复原脚本的命令，从目录分区开始。

  或者，可以先创建一个与目标数据库具有相同名称的哑数据库。在创建了哑数据库之后，可以在所有数据库分区上同时生成重定向复原脚本。
- 在发出 RESTORE 命令以生成脚本时，即使指定了 REPLACE EXISTING 选项，脚本中出现的 REPLACE EXISTING 选项也会被注释掉。
- 为了提高安全性，密码不会出现在生成的脚本中。您需要手动填写密码。
- 不能使用控制中心中的“复原”向导来生成重定向复原脚本。

要使用脚本来执行重定向复原:

1. 使用 restore 实用程序来生成重定向复原脚本。可以通过命令行处理器（CLP）或 db2Restore 应用程序编程接口（API）来调用 restore 实用程序。以下是指定了 REDIRECT 选项和 GENERATE SCRIPT 选项的 RESTORE DATABASE 命令示例:

   ```
   db2 restore db test from /home/jseifert/backups taken at 20050304090733
               redirect generate script test_node0000.clp
   ```

   此命令将在客户机上创建名为 `test_node0000.clp` 的重定向复原脚本。
2. 在文本编辑器中打开重定向复原脚本以进行所需的修改。可以修改:
   - 复原选项
   - 自动存储路径
   - 容器布局和路径
3. 运行修改后的重定向复原脚本。例如:

   ```
   db2 -tvf test_node0000.clp
   ```

RESTORE DATABASE

RESTORE DATABASE 命令重新创建已损坏或毁坏的数据库，该数据库已使用 DB2 BACKUP 实用程序进行备份。复原的数据库与创建备份副本时所处的状态相同。此实用程序还可以用另一映像来覆盖数据库，或将备份副本复原到新数据库。

有关不同操作系统和硬件平台之间的 DB2 数据库系统支持的复原操作的信息，请参阅《*数据恢复与高可用性指南和参考*》中的『不同操作系统和硬件平台之间的备份与复原操作』。

另外，还可使用 RESTORE 实用程序来复原在 DB2 通用数据库版本 8 上生成的备份映像。如果需要迁移，将在复原操作结束时自动调用该迁移。

如果在备份操作时启用了数据库以进行前滚恢复，可在成功完成复原操作后调用 ROLLFORWARD 实用程序将该数据库的状态恢复为以前的状态。

此实用程序还可复原表空间级备份。

当操作系统或字大小（32 位或 64 位）存在差别时，不能复原增量映像或仅捕获与先前映像的差别的映像（称为“差异映像”）。

在从一个环境成功复原到另一个环境后，在进行非增量备份前，不允许进行增量备份或差异备份。（对于在相同环境中执行复原操作的情况，那么不存在这样的局限性。）

即使从一个环境成功复原到另一个环境中，也需要注意：程序包在使用前必须重新绑定（使用 BIND 命令、REBIND 命令或 db2rbind 实用程序）；必须删除并重新创建 SQL 过程；必须在新平台上重构所有外部库。（对于复原到同一环境的情况，那么不存在这样的注意事项。）

作用域

此命令只影响对其执行该命令的节点。

权限

要复原到现有数据库，授予下列其中一个权限：

- *sysadm*
- *sysctrl*
- *sysmaint*

要复原到新数据库，授予下列其中一个权限：

- *sysadm*
- *sysctrl*

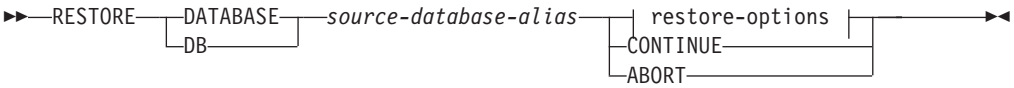
必需的连接

视复原操作类型的不同，所需连接也会有所不同：

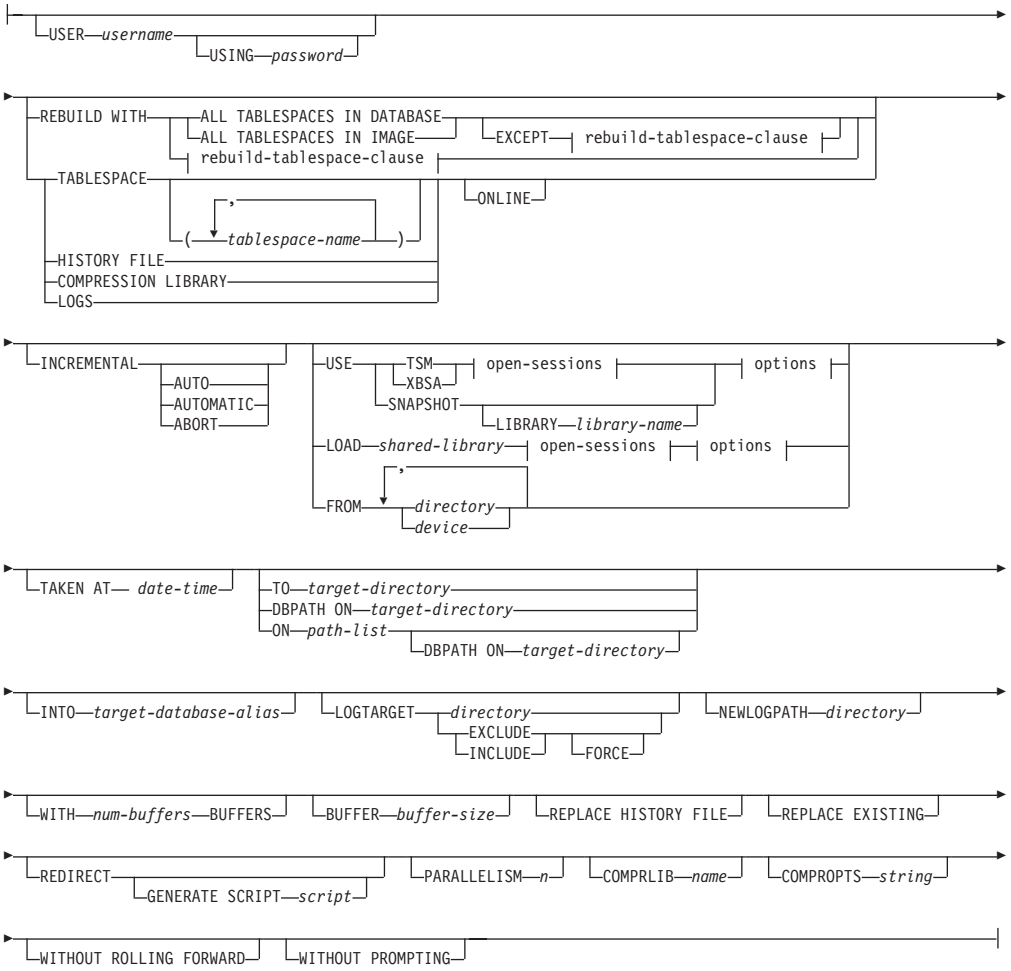
- 要复原到现有数据库，需要数据库连接。此命令自动建立与指定数据库的互斥连接。
- 要复原到新数据库，需要实例和数据库连接。创建数据库需要实例连接。

  要复原到不同于当前实例的实例上的新数据库，必须首先与新数据库将驻留的实例相连接。新实例可为本地实例或远程实例。当前实例是由 DB2INSTANCE 环境变量的值定义的。
- 要进行快照复原，需要*实例*和*数据库*连接。

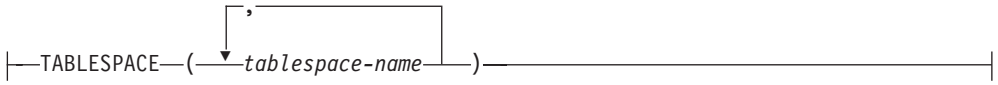
命令语法



restore-options:



rebuild-tablespace-clause:



open-sessions:

options:

命令参数

DATABASE *source-database-alias*
根据其建立备份的源数据库的别名。

CONTINUE
指定已重新定义了容器，且应执行“重定向复原”操作中的最后一步。

ABORT
此参数:

- 停止“重定向复原”操作。当发生需要重复一个或多个步骤的错误时，此参数很有用。发出指定了 ABORT 选项的 RESTORE DATABASE 后，必须重复“重定向复原”操作的每一步，包括指定了 REDIRECT 选项的 RESTORE DATABASE。
- 在增量复原操作完成前将其终止。

USER *username*
标识复原数据库时要使用的用户名。

USING *password*
用于认证用户名的密码。如果省略了密码，那么将提示用户输入。

REBUILD WITH ALL TABLESPACES IN DATABASE
在复原映像时，还将复原数据库及其已知的所有表空间。如果数据库已存在，那么此复原将覆盖该数据库。

REBUILD WITH ALL TABLESPACES IN DATABASE EXCEPT *rebuild-tablespace-clause*
在复原映像时，还将复原数据库及其已知的所有表空间，列表中指定的数据库除外。如果数据库已存在，那么此复原将覆盖该数据库。

REBUILD WITH ALL TABLESPACES IN IMAGE
复原要复原的映像中仅带有表空间的数据库。如果数据库已存在，那么此复原将覆盖该数据库。

REBUILD WITH ALL TABLESPACES IN IMAGE EXCEPT *rebuild-tablespace-clause*
复原要复原的映像中仅带有表空间的数据库，列表中指定的数据库除外。如果数据库已存在，那么此复原将覆盖该数据库。

REBUILD WITH *rebuild-tablespace-clause*
复原仅带有一组指定的表空间的数据库。如果数据库已存在，那么此复原将覆盖该数据库。

TABLESPACE *tablespace-name*
用于指定要复原的表空间的名称列表。

ONLINE
此关键字只可在执行表空间级的复原操作时应用。指定该关键字可联机复原备份映像。这意味着在对备份映像进行复原时，其他代理进程可连接至数据库，而复原指定的表空间时，其他表空间中的数据将是可用的。

HISTORY FILE
指定此关键字，以从备份映像中只复原历史记录文件。

COMPRESSION LIBRARY
指定此关键字，以从备份映像中只复原压缩库。如果对象存在于备份映像中，它将被复原到数据库目录中。如果对象不存在于备份映像中，那么复原操作将失败。

LOGS 指定此关键字，只复原备份映像中包含的日志文件的集合。如果备份映像不包含任何日志文件，那么复原操作将失败。如果指定了此选项，那么还必须指定 LOGTARGET 选项。

INCREMENTAL
不需要其他参数，使用 INCREMENTAL 即可指定手动累积复原操作。在手动复原期间，用户必须对复原涉及的每个映像手动发出每个复原命令。按照以下顺序完成此操作：最后一个、第一个、第二个、第三个，以此类推，并且包括最后一个映像。

INCREMENTAL AUTOMATIC/AUTO
指定自动累积复原操作。

INCREMENTAL ABORT
指定正在进行的手动累积复原操作异常中止。

USE

TSM 指定将从 Tivoli Storage Manager 管理的输出复原数据库。

XBSA 指定将使用 XBSA 接口。“备份服务 API”（XBSA）是一个开放的应用程序编程接口，由需要数据存储管理的应用程序或工具进行备份或归档时使用。

SNAPSHOT
指定将从快照备份复原数据。

不能将 SNAPSHOT 参数与以下任一参数配合使用:

- INCREMENTAL
- TO
- ON
- DBPATH ON
- INTO
- NEWLOGPATH
- WITH *num-buffers* BUFFERS
- BUFFER
- REDIRECT
- REPLACE HISTORY FILE
- COMPRESSION LIBRARY
- PARALLELISM
- COMPRLIB
- OPEN *num-sessions* SESSIONS
- HISTORY FILE

- LOGS

另外，不能将 SNAPSHOT 参数与任何涉及到表空间列表的复原操作（包括 REBUILD WITH 选项）一起使用。

如果未提供时间戳记（INCLUDE LOGS 是所有快照备份的缺省参数，除非显式声明了 EXCLUDE LOGS），那么从快照备份映像复原数据时的缺省行为是对构成数据库（包括所有容器）的所有路径、本地卷目录、数据库路径（DBPATH）以及最新快照备份的主日志和镜像日志路径执行 FULL DATABASE OFFLINE 复原。如果提供了时间戳记，那么将复原该快照备份映像。

LIBRARY *library-name*

集成到 IBM 数据服务器的是适用于下列存储器硬件的 DB2 ACS API 驱动程序:

- IBM TotalStorage® SAN Volume Controller
- IBM 企业存储服务器®型号 800
- IBM 系统存储器™ DS6000™
- IBM 系统存储器 DS8000™
- IBM 系统存储器 N 系列
- NetApp V 系列

如果您具有其他存储器硬件以及适用于该存储器硬件的 DB2 ACS API 驱动程序，那么您可以使用 LIBRARY 参数来指定 DB2 ACS API 驱动程序。

LIBRARY 参数的值是库的标准文件名。

OPTIONS

"options-string"

指定要用于复原操作的选项。该字符串将按输入时的形式（不带双引号）传递给 DB2 ACS API 驱动程序。不能使用 **VENDOROPT** 数据库配置参数对快照复原操作指定特定于供应商的选项。必须改用 RESTORE 实用程序的 OPTIONS 参数。

@*file-name*

指定 DB2 服务器上的某个文件中包含将用于复原操作的选项。会将该字符串传递给供应商支持库。该文件必须是标准文件名。

OPEN *num-sessions* **SESSIONS**

指定将与 TSM 或供应商产品一起使用的 I/O 会话的个数。

FROM *directory/device*

备份映像所在的目录或设备的标准路径名。如果省略了 USE TSM、FROM 和 LOAD，那么缺省值为客户机的当前工作目录。此目标目录或设备必须存在于目标服务器/实例上。

如果指定了若干项，而最后一项是磁带设备，那么将提示用户放入另一磁带。有效的响应选项为:

c 继续 - 继续使用生成了警告消息的设备（例如，在装上新磁带后继续）。

d 设备终止 - 只停止使用生成了警告消息的设备（例如，在没有更多磁带时终止）。

t 终止 - 在用户执行某些由实用程序请求的操作失败后，异常终止复原操作。

LOAD *shared-library*

共享库（在 Windows 操作系统上为 DLL）的名称，该共享库包含要使用的供应商备份与复原 I/O 函数。该名称可包含完整路径。如果未提供完整路径，该值将缺省为用户出口程序所在的路径。

TAKEN AT *date-time*

数据库备份映像的时间戳记。时间戳记在成功完成备份操作后显示，并且是备份映像的路径名的一部分。以格式 *yyyymmddhhmmss* 指定。还可指定部分时间戳记。例如，如果存在时间戳记分别为 20021001010101 和 20021002010101 的两个不同备份映像，那么指定 `20021002` 将会导致使用时间戳记为 `20021002010101` 的映像。如果未指定此参数的值，源介质上必须只有一个备份映像。

TO *target-directory*

此参数表示目标数据库目录。如果实用程序复原到一个现有数据库，将忽略此参数。指定的驱动器和目录必须是本地的。如果备份映像包括启用了自动存储器的数据库，那么仅数据库目录被更改，而与数据库关联的存储路径不更改。

DBPATH ON *target-directory*

此参数表示目标数据库目录。如果实用程序复原到一个现有数据库，将忽略此参数。指定的驱动器和目录必须是本地的。如果备份映像包括启用了自动存储器的数据库，且未指定 ON 参数，那么此参数与 TO 参数是同义词，且仅数据库目录被更改，而与数据库关联的存储路径不更改。

ON *path-list*

此参数重新定义与自动存储器数据库相关联的存储路径。将此参数与未启用自动存储器的数据库将导致错误（SQL20321N）。不再使用备份映像中定义的现有存储路径，自动存储器表空间自动重定向至新路径。如果没有为自动存储器数据库指定此参数，那么存储路径仍为备份映像中定义的路径。

可指定一个或多个路径，各个路径之间用逗号分隔。每个路径必须有绝对路径名称且该路径必须在本地。如果磁盘上尚没有该数据库，且未指定 DBPATH ON 参数，那么第一个路径用作目标数据库目录。

对于多分区数据库，只能在目录分区上指定 ON *path-list* 选项。使用 ON 选项时，必须先复原目录分区然后再复原其他分区。使用新存储路径复原目录分区会将所有非目录节点置于 RESTORE_PENDING 状态。这样，非目录节点就可并行复原，而无需在复原命令中指定 ON 子句。

通常，必须为多分区数据库中的每个分区使用相同的存储路径，且在执行 RESTORE DATABASE 命令前，这些分区必须都存在。唯一的例外情况是在存储路径中使用了数据库分区表达式。使用数据库分区表达式作为存储路径名的一部分允许在存储路径中反映数据库分区号，这样生成的路径名在每个分区上都不相同。

使用自变量 `" $N" ([blank]$N)` 来指示数据库分区表达式。数据库分区表达式可以用在存储路径中的任何位置，并且可指定多个数据库分区表达式。使用空格字符来表示数据库分区表达式的结束区表达式以空格字符结束；在对数据库

分区表达式求值以后，该空格后跟的所有内容都将追加至存储路径。如果存储路径中的数据库分区表达式后面没有空格字符，那么假定其余字符串是该表达式的一部分。只能用下列其中一种格式使用该自变量:

表 49. . 从左往右对运算符求值。% 表示模数运算符。假定示例中的数据库分区号为 10。

语法	示例	值
[blank]$N	″ $N″	10
[blank]$N+[number]	″ $N+100″	110
[blank]$N%[number]	″ $N%5″	0
[blank]$N+[number]%[number]	″ $N+1%5″	1
[blank]$N%[number]+[number]	″ $N%4+2″	4
[a] % 是模数。		

INTO *target-database-alias*

目标数据库别名。如果目标数据库不存在，那么创建。

将数据库备份复原到现有数据库时，复原的数据库继承现有数据库的别名和数据库名称。将数据库备份复原到不存在的数据库时，使用您指定的别名和数据库名称创建新数据库。此新数据库名称在复原它的系统上必须唯一。

LOGTARGET *directory*

非快照复原:

数据库服务器上现有目录的绝对路径名，此路径名作为目标目录，用于从备份映像中抽取日志文件。如果指定该选项，那么将备份映像中包含的任何日志文件抽取至目标目录。如果未指定该选项，将不抽取备份映像中包含的日志文件。要从备份映像中仅抽取日志文件，请指定 LOGS 选项。

快照复原:

INCLUDE

从快照映像复原日志目录卷。如果指定了此选项且备份映像包含日志目录，那么将复原这些日志目录。磁盘上的现有日志目录和日志文件如果与备份映像中的日志目录不发生冲突，那么将保持不变。如果磁盘上的现有日志目录与备份映像中的日志目录冲突，那么将返回错误。

EXCLUDE

不复原日志目录卷。如果指定了此选项，那么将不从备份映像复原任何日志目录。磁盘上的现有日志目录和日志文件如果与备份映像中的日志目录不发生冲突，那么将保持不变。如果复原了属于数据库的路径，并且将因此隐式复原了日志目录，从而导致日志目录被覆盖，那么将返回错误。

FORCE

在复原快照映像时，允许覆盖和替换当前数据库中的现有日志目录。如果未指定此选项，那么当磁盘上的现有日志目录和日志文件与快照映像中的日志目录冲突时，将导致复原失败。使用此选项来指示复原操作可以覆盖和替换那些现有的日志目录。

注: 使用此选项时应格外小心，并且务必确保备份和归档了执行恢复时所必需的所有日志。

注: 如果未指定 LOGTARGET，那么缺省值为 LOGTARGET EXCLUDE。

NEWLOGPATH *directory*

在复原操作后用于活动日志文件的目录的绝对路径名。此参数与数据库配置参数 **newlogpath** 有相同的功能，但此参数的影响只限于它所指定的复原操作。当备份映像中的日志路径不适合在复原操作后使用时可使用此参数，例如，在该路径不再有效时或由另一数据库使用时。

WITH *num-buffers* **BUFFERS**

将要使用的缓冲区数量。除非您显式地输入一个值，否则 DB2 数据库系统将自动为此参数选择最佳值。从多个源读取数据时，或增加了 PARALLELISM 的值时，可使用较大的缓冲区数以改进性能。

BUFFER *buffer-size*

将用于复原操作的缓冲区大小（以页计）。除非您显式地输入一个值，否则 DB2 数据库系统将自动为此参数选择最佳值。此参数的最小值是 8 页。

复原缓冲区大小必须是在备份操作期间指定的备份缓冲区大小的正整数倍数。如果指定了不正确的缓冲区大小，那么缓冲区将分配为最小的可接受大小。

REPLACE HISTORY FILE

指定复原操作应该使用备份映像中的历史记录文件替换磁盘上的历史记录文件。

REPLACE EXISTING

如果存在一个别名与目标数据库别名相同的数据库，此参数指定 RESTORE 实用程序将使用复原数据库替换现有数据库。此参数对于调用 RESTORE 实用程序的脚本很有用，这是因为命令行处理器将不会提示用户验证是否删除了现有数据库。如果指定了 WITHOUT PROMPTING 参数，那么不需要指定 REPLACE EXISTING，但在这种情况下，如果发生了通常需要用户干预的事件，操作会失败。

REDIRECT

指定“重定向复原”操作。要完成“重定向复原”操作，在此命令后应跟有一个或多个 SET TABLESPACE CONTAINERS 命令，然后跟有指定了 CONTINUE 选项的 RESTORE DATABASE 命令。必须从同一窗口或 CLP 会话调用与一个“重定向复原”操作相关的所有命令。不能对启用自动存储器的表空间执行“重定向复原”操作。

GENERATE SCRIPT *script*

使用指定的文件名创建重定向复原脚本。脚本名称可以是相对的或绝对的，脚本将在客户端生成。如果不能在客户端创建文件，将返回错误消息（SQL9304N）。如果文件已存在，那么将覆盖存在的文件。请参阅以下示例以了解进一步的用法信息。

WITHOUT ROLLING FORWARD

指定在成功复原数据库后，不要将该数据库置于前滚暂挂状态。

如果成功完成复原操作后，该数据库处于前滚暂挂状态，那么必须调用 ROLLFORWARD 命令，才能再次使用该数据库。

从联机备份映像复原时，如果指定此选项，那么将返回错误 SQL2537N。

如果备份映像是一个可恢复的数据库，那么 WITHOUT ROLLING FORWARD 不能指定为带有 REBUILD 选项。

PARALLELISM *n*

指定将在复原操作期间创建的缓冲区操纵程序的数量。除非您显式地输入一个值，否则 DB2 数据库系统将自动为此参数选择最佳值。

COMPRLIB *name*

指示要用于执行解压的库的名称（例如，对于 Windows，为 db2compr.dll；对于 Linux/UNIX 系统，为 libdb2compr.so）。此名称必须是引用服务器上某个文件的标准路径。如果未指定此参数，那么 DB2 将尝试使用映像中存储的库。如果备份未压缩，那么将忽略此参数的值。如果无法装入指定的库，那么复原操作将失败。

COMPROPTS *string*

描述传递给解压库中的初始化例程的二进制数据块。DB2 数据库系统将此字符串从客户机直接传递给服务器，因此解压库将处理所有字节逆转或代码页转换问题。如果数据块的第一个字符是“@”，那么 DB2 数据库系统会将数据的其他部分解释为服务器上某个文件的名称。DB2 数据库系统随后会将用此文件的内容替换 *string* 的内容，并将此新值传递给初始化例程。string 的最大长度为 1 024 字节。

WITHOUT PROMPTING

指定复原操作将以无人照管方式运行。那些通常需要用户干预的操作将返回一条错误消息。如果使用可移动介质设备（例如，磁带或软盘），即使指定此选项也会在设备结束时提示用户。

示例

1. 在以下示例中，数据库 WSDB 是在编号为 0 至 3 的这 4 个数据库分区上定义的。可以从所有数据库分区访问路径 /dev3/backup。可以从 /dev3/backup 获取下列脱机备份映像:

```
wsdb.0.db2inst1.NODE0000.CATN0000.20020331234149.001
wsdb.0.db2inst1.NODE0001.CATN0000.20020331234427.001
wsdb.0.db2inst1.NODE0002.CATN0000.20020331234828.001
wsdb.0.db2inst1.NODE0003.CATN0000.20020331235235.001
```

要先复原目录分区，然后从 /dev3/backup 目录复原 WSDB 数据库的所有其他数据库分区，从其中一个数据库分区发出下列命令:

```
db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
  INTO wsdb REPLACE EXISTING'
db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
  INTO wsdb REPLACE EXISTING'
db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
  INTO wsdb REPLACE EXISTING'
db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
  INTO wsdb REPLACE EXISTING'
```

db2_all 实用程序将对每个指定的数据库分区发出复原命令。使用 db2_all 执行复原时，应始终指定 REPLACE EXISTING 和/或 WITHOUT PROMPTING。否则，如果有提示，那么该操作看起来像挂起一样。这是因为 db2_all 不支持用户提示。

2. 以下是一个典型的重定向复原方案，用于别名为 MYDB 的数据库:
   a. 发出 RESTORE DATABASE 命令，请使用 REDIRECT 选项。

```
restore db mydb replace existing redirect
```

成功完成步骤 1 之后且在完成步骤 3 之前，通过发出以下命令来中止复原操作:

```
restore db mydb abort
```

b. 对必须重新定义容器的每个表空间发出 SET TABLESPACE CONTAINERS 命令。例如:

```
set tablespace containers for 5 using
   (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
```

要验证复原数据库的容器是否是在此步骤中指定的那些容器，请发出 LIST TABLESPACE CONTAINERS 命令。

c. 在成功完成步骤 1 和 2 后，请发出:

```
restore db mydb continue
```

这是“重定向复原”操作的最后一步。

d. 如果步骤 3 失败，或者如果中止了复原操作，那么可从步骤 1 开始重新启动重定向的复原。

3. 以下是对可恢复数据库每周进行的增量备份策略的样本。包括一个每周进行的完整数据库备份操作、一个每天进行的非累积（差异）备份操作以及一个在每周中期进行的累积（增量）备份操作:

```
(Sun) backup db mydb use tsm
(Mon) backup db mydb online incremental delta use tsm
(Tue) backup db mydb online incremental delta use tsm
(Wed) backup db mydb online incremental use tsm
(Thu) backup db mydb online incremental delta use tsm
(Fri) backup db mydb online incremental delta use tsm
(Sat) backup db mydb online incremental use tsm
```

要对在星期五早上创建的映像自动进行数据库复原，请发出:

```
restore db mydb incremental automatic taken at (Fri)
```

要对在星期五早上创建的映像手动进行数据库复原，请发出:

```
restore db mydb incremental taken at (Fri)
restore db mydb incremental taken at (Sun)
restore db mydb incremental taken at (Wed)
restore db mydb incremental taken at (Thu)
restore db mydb incremental taken at (Fri)
```

4. 要产生包括日志的备份映像，以传送至远程站点:

```
backup db sample online to /dev3/backup include logs
```

要复原备份映像，请提供 LOGTARGET 路径并在 ROLLFORWARD 期间指定此路径:

```
restore db sample from /dev3/backup logtarget /dev3/logs
rollforward db sample to end of logs and stop overflow log path /dev3/logs
```

5. 要仅从包含日志的备份映像中检索日志文件:

```
restore db sample logs from /dev3/backup logtarget /dev3/logs
```

6. 可以使用 USE TSM OPTIONS 关键字指定 TSM 信息，以供复原操作使用。在 Windows 平台上，省略 -fromowner 选项。

- 指定定界字符串:

```
restore db sample use TSM options '"-fromnode=bar -fromowner=dmcinnis"'
```

- 指定标准文件:

```
restore db sample use TSM options @/u/dmcinnis/myoptions.txt
```

文件 myoptions.txt 包含下列信息: -fromnode=bar -fromowner=dmcinnis

7. 以下是一个使用新存储路径来简单复原启用了多分区自动存储器的数据库的示例。数据库最初使用一个存储路径 /myPath0: 来创建
   - 在目录分区上发出: restore db mydb on /myPath1,/myPath2
   - 在所有非目录分区上发出: restore db mydb
8. 非自动存储器数据库上以下命令的脚本输出:

```
restore db sample from /home/jseifert/backups taken at 20050301100417 redirect
generate script SAMPLE_NODE0000.clp
```

将类似于以下示例:

```
-- ****************************************************************************
-- ** 自动创建的重定向复原脚本
-- ****************************************************************************
UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM  0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- ****************************************************************************
-- ** 初始化重定向复原
-- ****************************************************************************
   RESTORE DATABASE SAMPLE
-- USER  '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050301100417
-- DBPATH ON '<target-directory>'
INTO SAMPLE
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/SQLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- ****************************************************************************
-- ** 表空间定义
-- ****************************************************************************
-- ****************************************************************************
-- ** 表空间名称                          = SYSCATSPACE
-- **   表空间标识                        = 0
-- **   表空间类型                        = 系统管理空间
-- **   表空间内容类型                    = 任意数据
-- **   表空间页大小（字节）              = 4096
-- **   表空间扩展数据块大小（页）        = 32
-- **   使用自动存储器                    = No
-- **   总页数                              = 5572
-- ****************************************************************************
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0000.0'
);
-- ****************************************************************************
-- ** 表空间名称                          = TEMPSPACE1
-- **   表空间标识                        = 1
-- **   表空间类型                        = 系统管理空间
-- **   表空间内容类型                    = 系统临时数据
-- **   表空间页大小（字节）              = 4096
```

```
-- **   表空间扩展数据块大小（页）         = 32
-- **   使用自动存储器                     = No
-- **   总页数                             = 0
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0001.0'
);
-- ******************************************************************************
-- ** 表空间名称                         = USERSPACE1
-- **   表空间标识                         = 2
-- **   表空间类型                         = 系统管理空间
-- **   表空间内容类型                     = 任意数据
-- **   表空间页大小（字节）               = 4096
-- **   表空间扩展数据块大小（页）         = 32
-- **   使用自动存储器                     = No
-- **   总页数                            = 1
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH   'SQLT0002.0'
);
-- ******************************************************************************
-- ** 表空间名称                         = DMS
-- **   表空间标识                         = 3
-- **   表空间类型                         = 数据库管理空间
-- **   表空间内容类型                     = 任意数据
-- **   表空间页大小（字节）               = 4096
-- **   表空间扩展数据块大小（页）         = 32
-- **   使用自动存储器                     = No
-- **   启用自动调整大小                   = No
-- **   总页数                             = 2000
-- **   可用页数                           = 1960
-- **   高水位标记（页）                   = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   /tmp/dms1                                                    1000
, FILE   /tmp/dms2                                                    1000
);
-- ******************************************************************************
-- ** 表空间名称                         = RAW
-- **   表空间标识                         = 4
-- **   表空间类型                         = 数据库管理空间
-- **   表空间内容类型                     = 任意数据
-- **   表空间页大小（字节）               = 4096
-- **   表空间扩展数据块大小（页）         = 32
-- **   使用自动存储器                     = No
-- **   启用自动调整大小                   = No
-- **   总页数                             = 2000
-- **   可用页数                           = 1960
-- **   高水位标记（页）                   = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                          1000
, DEVICE '/dev/hdb2'                                          1000
);
-- ******************************************************************************
-- ** 启动重定向复原
-- ******************************************************************************
   RESTORE DATABASE SAMPLE CONTINUE;
```

```
-- ********************************************************************************
-- ** 文件结束
-- ********************************************************************************
```

9. 自动存储器数据库上以下命令的脚本输出:

```
restore db test from /home/jseifert/backups taken at 20050304090733 redirect
generate script TEST_NODE0000.clp
```

将类似于以下示例:

```
-- ********************************************************************************
-- ** 自动创建的重定向复原脚本
-- ********************************************************************************
UPDATE COMMAND OPTIONS USING S ON Z ON TEST_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM  0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- ********************************************************************************
-- ** 初始化重定向复原
-- ********************************************************************************
RESTORE DATABASE TEST
-- USER  '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050304090733
ON '/home/jseifert'
-- DBPATH ON <target-directory>
INTO TEST
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00002/SQLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- ********************************************************************************
-- ** 表空间定义
-- ********************************************************************************
-- ********************************************************************************
-- ** 表空间名称                              = SYSCATSPACE
-- **   表空间标识                            = 0
-- **   表空间类型                            = 数据库管理空间
-- **   表空间内容类型                        = 任意数据
-- **   表空间页大小（字节）                  = 4096
-- **   表空间扩展数据块大小（页）            = 4
-- **   使用自动存储器                        = Yes
-- **   启用自动调整大小                      = Yes
-- **   总页数                                = 6144
-- **   可用页数                              = 6140
-- **   高水位标记（页）                      = 5968
-- ********************************************************************************
-- ********************************************************************************
-- ** 表空间名称                              = TEMPSPACE1
-- **   表空间标识                            = 1
-- **   表空间类型                            = 系统管理空间
-- **   表空间内容类型                        = 系统临时数据
-- **   表空间页大小（字节）                  = 4096
-- **   表空间扩展数据块大小（页）            = 32
-- **   使用自动存储器                        = Yes
-- **   总页数                                = 0
-- ********************************************************************************
-- ********************************************************************************
-- ** 表空间名称                              = USERSPACE1
-- **   表空间标识                            = 2
-- **   表空间类型                            = 数据库管理空间
```

```
-- **   表空间内容类型                    = 任意数据
-- **   表空间页大小（字节）              = 4096
-- **   表空间扩展数据块大小（页）        = 32
-- **   使用自动存储器                    = Yes
-- **   启用自动调整大小                  = Yes
-- **   总页数                            = 256
-- **   可用页数                          = 224
-- **   高水位标记（页）                  = 96
-- ******************************************************************************
-- ******************************************************************************
-- ** 表空间名称                         = DMS
-- **   表空间标识                        = 3
-- **   表空间类型                        = 数据库管理空间
-- **   表空间内容类型                    = 任意数据
-- **   表空间页大小（字节）              = 4096
-- **   表空间扩展数据块大小（页）        = 32
-- **   使用自动存储器                    = No
-- **   启用自动调整大小                  = No
-- **   总页数                            = 2000
-- **   可用页数                          = 1960
-- **   高水位标记（页）                  = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE   '/tmp/dms1'                                          1000
, FILE   '/tmp/dms2'                                          1000
);
-- ******************************************************************************
-- ** 表空间名称                         = RAW
-- **   表空间标识                        = 4
-- **   表空间类型                        = 数据库管理空间
-- **   表空间内容类型                    = 任意数据
-- **   表空间页大小（字节）              = 4096
-- **   表空间扩展数据块大小（页）        = 32
-- **   使用自动存储器                    = No
-- **   启用自动调整大小                  = No
-- **   总页数                            = 2000
-- **   可用页数                          = 1960
-- **   高水位标记（页）                  = 96
-- ******************************************************************************
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1'                                          1000
, DEVICE '/dev/hdb2'                                          1000
);
-- ******************************************************************************
-- ** 启动重定向复原
-- ******************************************************************************
RESTORE DATABASE TEST CONTINUE;
-- ******************************************************************************
-- ** 文件结束
-- ******************************************************************************
```

10. 以下是使用 SNAPSHOT 选项的 RESTORE DB 命令示例:

    从快照映像复原日志目录卷且不提示。

    ```
    db2 restore db sample use snapshot LOGTARGET INCLUDE without prompting
    ```

    不复原日志目录卷且不提示。

    ```
    db2 restore db sample use snapshot LOGTARGET EXCLUDE without prompting
    ```

    不复原日志目录卷且不提示。如果未指定 LOGTARGET，那么缺省值为 LOGTARGET EXCLUDE。

```
db2 restore db sample use snapshot without prompting
```

在复原包含相冲突的日志目录的快照映像时，允许覆盖和替换当前数据库中的现有日志目录，但不提示。

```
db2 restore db sample use snapshot LOGTARGET EXCLUDE FORCE without prompting
```

在复原包含相冲突的日志目录的快照映像时，允许覆盖和替换当前数据库中的现有日志目录，但不提示。

```
db2 restore db sample use snapshot LOGTARGET INCLUDE FORCE without prompting
```

使用说明

- 格式为 db2 restore db <name> 的 RESTORE DATABASE 命令将使用数据库映像执行整个数据库复原，并且将对表空间映像中的表空间执行表空间复原操作。格式为 db2 restore db <name> tablespace 的 RESTORE DATABASE 命令对映像中的表空间执行表空间复原。另外，如果使用此命令时提供了一组表空间，那么复原这些显式列示的表空间。
- 在进行联机备份的复原操作后，必须执行前滚恢复。
- 如果备份映像是压缩的，那么 DB2 数据库系统会检测到这种情况，并在复原数据前自动解压数据。如果在 db2Restore API 上指定库，那么将使用该库来解压数据。否则，将进行检查以确定是否在备份映像中存储了库、库是否存在以及是否已使用。最后，如果在备份映像中没有存储库，那么不能解压数据，复原操作失败。
- 如果要从备份映像复原压缩库（通过显式地指定 COMPRESSION LIBRARY 选项或通过隐式执行压缩备份的正常复原），那么必须在执行备份的平台和操作系统上执行复原操作。如果执行备份所用的平台与执行复原所用的平台不同，那么复原操作将失败，即使 DB2 正常支持涉及两个系统的跨平台复原亦如此。
- 要从包含日志文件的备份映像中复原这些日志文件，必须指定 LOGTARGET 选项，并提供 DB2 服务器上存在的有效标准路径。如果这些条件都满足，那么 RESTORE 实用程序将日志文件从映像中写入目标路径。如果在不包括日志的备份映像复原期间指定 LOGTARGET，那么复原操作将返回错误，然后再尝试复原任何表空间数据。如果指定无效或只读的 LOGTARGET 路径，那么复原操作也会失败并返回错误。
- 如果在发出 RESTORE DATABASE 命令时，LOGTARGET 路径中存在任何日志文件，那么将向用户返回一条警告提示。如果指定了 WITHOUT PROMPTING，那么将不返回此警告。
- 在指定 LOGTARGET 的复原操作期间，如果无法抽取任何日志文件，那么复原操作将失败并返回错误。如果正在从备份映像中抽取的任何日志文件与 LOGTARGET 路径中现有的文件同名，那么复原操作将失败，并返回错误。复原数据库实用程序不会覆盖 LOGTARGET 目录中的现有日志文件。
- 还可以从备份映像中仅复原保存的日志集。要指示仅复原日志文件，除 LOGTARGET 路径之外，还请指定 LOGS 选项。指定不带 LOGTARGET 路径的 LOGS 选项将导致错误。如果以此操作方式复原日志文件时发生任何问题，复原操作将立即终止并将返回错误。
- 自动增量复原操作期间，仅从备份映像中检索复原操作目标映像中包含的日志文件。不会从那些中间备份映像中抽取中间映像（在增量复原操作处理期间引用）包含的任何日志文件。手动增量复原操作期间，只应通过发出最终复原命令来指定 LOGTARGET 路径。
- 数据库脱机完全备份和数据库脱机增量备份可以复原为更高的数据库版本，而联机备份则不能。对于多分区数据库，必须先单独地复原目录分区，然后再复原其余数

据库分区（以并行或串行方式）。然而，复原操作所执行的隐式数据库迁移可能失败。在多分区数据库中，隐式数据库迁移可能在一个或多个数据库分区上失败。在这种情况下，通过从目录分区中发出 RESTORE DATABASE 命令，然后接着发出单个 MIGRATE DATABASE 命令可以成功迁移数据库。

快照复原

类似传统（非快照）复原，复原快照备份映像时的缺省行为将不复原日志目录 - LOGTARGET EXCLUDE。

如果 DB2 管理器检测到在要复原的任何其他路径中共享了任何日志目录的组标识，那么将返回错误。在这种情况下，必须指定 LOGTARGET INCLUDE 或 LOGTARGET INCLUDE FORCE，这是因为日志目录必须是复原操作的一部分。

从备份映像复原路径前，DB2 管理器将设法保存现有的日志目录（主日志目录、镜像日志目录和溢出日志目录）。

如果您要复原日志目录且 DB2 管理器检测到磁盘上预先存在的日志目录与备份映像中的日志目录相冲突，那么 DB2 管理器将报告错误。在此类情况下，如果您指定了 LOGTARGET INCLUDE FORCE，那么将禁止此错误，然后预先删除任何已存在的日志目录并从映像复原日志目录。

存在一种特殊情况：指定了 LOGTARGET EXCLUDE 选项且日志目录路径位于数据库目录（例如 /NODExxxx/SQLxxxxx/SQLOGDIR/）下。在这种情况下，复原操作仍将覆盖该日志目录，这是因为将复原数据库路径及其下面的所有内容。如果 DB2 管理器检测到此情况且此日志目录中存在日志文件，那么将报告错误。如果您指定了 LOGTARGET EXCLUDE FORCE，那么将禁止此错误，并且备份映像中的那些日志目录将覆盖磁盘上相冲突的日志目录。

通过暂挂 I/O 和联机分割镜像支持获取高可用性

通过使用 IBM 数据服务器的暂挂 I/O 支持，可以分割主数据库的镜像副本，而不必使该数据库脱机。利用这种方法，可以在主数据库出现故障时迅速创建一个备用数据库来接管操作。

磁盘镜像是将数据同时写入两个单独的硬盘中的进程。一个数据副本是另一个数据副本的镜像。分割镜像是分离两个副本的进程。

可以使用磁盘镜像来维护主数据库的辅助副本。可以使用 IBM 数据服务器的暂挂 I/O 功能来分割数据库的主镜像副本和辅助镜像副本，而不必使该数据库脱机。分割主数据库副本和辅助数据库副本后，在主数据库出现故障时，辅助数据库可以接管操作。

如果不希望使用 IBM 数据服务器的 backup 实用程序来备份大型数据库，可通过使用暂挂 I/O 和分割镜像功能根据镜像映像来建立副本。此方法还可以:

- 消除来自生产机器的备份操作开销
- 提供了克隆系统的一个快捷方式
- 提供了对空闲备用故障转移的快速实现。不需要初始复原操作，如果证实前滚操作太慢或遇到了错误，重新初始化会非常快。

db2inidb 命令初始化分割镜像，因此可用来:

- 作为克隆数据库
- 作为备用数据库
- 作为备份映像

只能对分割镜像发出此命令，必须首先运行该命令才能使用分割镜像。

在分区数据库环境中，不必同时对所有数据库分区暂挂 I/O 写操作。可以暂挂由一个或多个数据库分区组成的数据库分区子集来创建用于执行脱机备份的分割镜像。如果该子集包括目录分区，那么它必须是要暂挂的最后一个数据库分区。

在分区数据库环境中，必须先对每个数据库分区运行 db2inidb 命令，然后才能使用根据任何这些数据库分区创建的分割映像。可以使用 db2_all 命令来同时对所有数据库分区运行此工具。然而，如果使用了 RELOCATE USING 选项，那么无法使用 db2_all 命令来同时对所有数据库分区运行 db2inidb。必须为每个数据库分区提供独立的配置文件，该配置文件包含所更改的数据库分区的 NODENUM 值。例如，如果要更改数据库的名称，这将影响每个数据库分区，并且必须在每个数据库分区上都有独立配置文件的情况下运行 db2relocatedb 命令。如果要移动属于单一数据库分区的容器，那么只需要在该数据库分区上运行一次 db2relocatedb 命令。

注: 确保分割镜像包含组成数据库的所有容器和目录（包括卷目录）。要收集此信息，请参阅 DBPATHS 管理视图，该视图显示了需要分割的数据库的所有文件和目录。

db2inidb - 初始化镜像数据库

在分割镜像环境中初始化镜像数据库。镜像数据库可以初始化为主数据库的克隆，并置于前滚暂挂状态或用作要复原主数据库的备份映像。只能对分割镜像数据库运行此命令，并且必须先运行此命令，然后才可以使用分割镜像。

权限

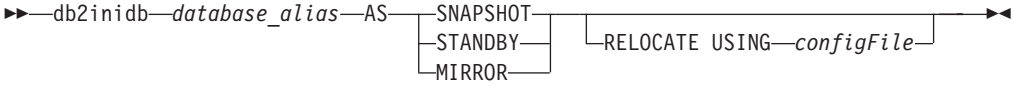
为下列其中一项:

- *sysadm*
- *sysctrl*
- *sysmaint*

必需的连接

无

命令语法



命令参数

database_alias
指定要初始化的数据库的别名。

SNAPSHOT
指定要初始化为主数据库的克隆的镜像数据库。

STANDBY
指定要置于前滚暂挂状态的数据库。可访存主数据库中的新日志，并应用到备用数据库。然后可在主数据库当机时，将该备用数据库代替主数据库。

MIRROR
指定要用作备份映像的镜像数据库，该备份映像可以用来复原主数据库。

RELOCATE USING *configFile*
指定在将数据库作为快照、备用或镜像数据库初始化前，根据指定的 *configFile* 中列示的信息重新定位数据库文件。在 『db2relocatedb - 重定位数据库』中描述了 *configFile* 的格式。

使用说明

发出 db2inidb *database_alias* as mirror 命令前，请不要发出 `db2 connect to` *database-alias* 操作。在初始化一个分割镜像数据库前与其进行连接，会删除前滚恢复期间需要的日志文件。此连接将数据库设置回它被暂挂时的状态。如果数据库暂挂期间被标记为一致，DB2 数据库系统将得出结论：没有必要进行崩溃恢复并清空日志以备将来使用。如果日志已经被清空，尝试前滚被返回的 SQL4970N 错误消息中的结果。

在分区数据库环境中，必须先对每个数据库分区运行 db2inidb，然后才能使用来自任何数据库分区的分割镜像。使用 db2_all 命令，可在所有数据库分区上同时运行 db2inidb。

然而，如果正在使用 RELOCATE USING 选项，那么无法使用 db2_all 命令在所有分区上同时运行 db2inidb。必须为每个分区提供独立的配置文件，该配置文件包含所更改的数据库分区的 NODENUM 值。例如，如果要更改数据库的名称，这将影响每个数据库分区，并且必须在每个数据库分区上都有独立配置文件的情况下运行 db2relocatedb 命令。如果要移动属于单一数据库分区的容器，那么只需要在该数据库分区上运行一次 db2relocatedb 命令。

如果指定了 RELOCATE USING *configFile* 参数且成功地重新定位了数据库，那么指定的 *configFile* 将被复制到数据库目录中且重命名为 `db2path.cfg`。在后续崩溃恢复或前滚恢复期间，当处理日志文件时使用此文件重命名容器路径。

如果正在初始化克隆数据库，那么在崩溃恢复完成后，指定的 *configFile* 将自动从数据库目录中除去。

如果正在初始化备用数据库或镜像数据库，那么在完成或取消前滚恢复后，指定的 *configFile* 将自动从数据库目录中除去。在运行 db2inidb 后，可将新容器路径添加至 `db2path.cfg` 文件。这在以下情况下是必要的：当在原始数据库上进行 CREATE 或 ALTER TABLESPACE 操作且必须在备用数据库上使用不同的路径时。

db2relocatedb - 重定位数据库

此命令将按照用户提供的配置文件中指定那样重命名数据库或者重定位数据库或数据库的一部分（例如，容器和日志目录）。此工具将对 DB2 实例和数据库支持文件进行必需的更改。

权限

无

命令语法

```
►►—db2relocatedb—-f— configFilename———————————————————————————►◄
```

命令参数

-f *configFilename*

指定包含重定位数据库时所需要的配置信息的文件的名称。此文件名可以是相对文件名或绝对文件名。配置文件的格式为:

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
...
STORAGE_PATH=oldStoragePath1,newStoragePath1
STORAGE_PATH=oldStoragePath2,newStoragePath2
...
```

其中:

DB_NAME

指定要重定位的数据库的名称。如果要更改数据库名称，那么必须同时指定旧名称和新名称。这是一个必需字段。

DB_PATH

指定要重定位的数据库的原始路径。如果要更改数据库路径，那么必须同时指定旧路径和新路径。这是一个必需字段。

INSTANCE

指定数据库所在的实例。如果要将数据库移至新实例，那么必须同时指定旧实例和新实例。这是一个必需字段。

NODENUM

指定要更改的数据库节点的节点号。缺省值为 0。

LOG_DIR

指定日志路径的位置的更改。如果要更改日志路径，那么必须同时指定旧路径和新路径。如果日志路径位于数据库路径下（在这种情况下将自动更新路径），那么此规范是可选的。

CONT_PATH

指定表空间容器的位置的更改。必须同时指定旧容器路径和新容器路径。如果要更改多个容器路径，那么可以提供多个 CONT_PATH 行。如果容器路径位于数据库路径下（在这种情况下将自动更新路径），那么此规范是可选的。如果要更改多个容器（这些容器的相同旧路径要被替换为一个公共新路径），那么可以使用单个 CONT_PATH 条目。在这种情况下，可以同时在旧路径和新路径中使用星号（*）作为通配符。

STORAGE_PATH

此参数仅适用于启用了自动存储器的数据库。它指定数据库的其中一个存储路径的位置的更改。必须同时指定旧存储路径和新存储路径。如果要更改多个存储路径，那么可以提供多个 STORAGE_PATH 行。

忽略了空白行或者以注释字符（#）开头的行。

示例

示例 1

要将位于 /home/db2inst1 路径上的 db2inst1 实例中的数据库名称 TESTDB更改为 PRODDB，那么应创建以下配置文件:

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

将该配置文件另存为 relocate.cfg，并使用以下命令来更改数据库文件:

```
db2relocatedb -f relocate.cfg
```

示例 2

要将 DATAB1 数据库从 /dbpath 路径上的 jsmith 实例移至 prodinst 实例，请执行以下操作:

1. 将 /dbpath/jsmith 目录中的文件移至 /dbpath/prodinst。
2. 将以下配置文件与 db2relocatedb 命令配合使用来更改数据库文件:

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

示例 3

PRODDB 数据库存在于 /databases/PRODDB 路径上的 inst1 实例中。需要按如下所示更改两个表空间容器的位置:

- 需要将 SMS 容器 /data/SMS1 移至 /DATA/NewSMS1。
- 需要将 DMS 容器 /data/DMS1 移至 /DATA/DMS1 。

将物理目录和文件移至新位置后，可以将以下配置文件与 db2relocatedb 命令配合使用来更改数据库文件，以便它们识别新位置:

```
DB_NAME=PRODDB
   DB_PATH=/databases/PRODDB
   INSTANCE=inst1
NODENUM=0
   CONT_PATH=/data/SMS1,/DATA/NewSMS1
   CONT_PATH=/data/DMS1,/DATA/DMS1
```

示例 4

数据库 TESTDB 存在于 db2inst1 实例中，并且是在 /databases/TESTDB 路径上创建的。然后在下列容器中创建了表空间:

```
TS1
   TS2_Cont0
   TS2_Cont1
   /databases/TESTDB/TS3_Cont0
   /databases/TESTDB/TS4/Cont0
   /Data/TS5_Cont0
   /dev/rTS5_Cont1
```

将 TESTDB 移至新系统。新系统上的实例将为 newinst，而数据库的位置将为 /DB2。

当移动数据库时，必须将 /databases/TESTDB/db2inst1 目录中存在的所有文件移至 /DB2/newinst 目录。这意味着前 5 个容器将被重定位为此移动操作的一部分。（前 3 个容器是相对于数据库目录的，后两个容器是相对于数据库路径的。）由于这些容器位于数据库目录或数据库路径中，因此不需要将它们列示在配置文件中。如果要将其余两个容器移至新系统上的其他位置，那么必须在配置文件中列示这两个容器。

将物理目录和文件移至它们的新位置后，可以将以下配置文件与 db2relocatedb 配合使用来更改数据库文件，以便它们识别新位置:

```
DB_NAME=TESTDB
DB_PATH=/databases/TESTDB,/DB2
INSTANCE=db2inst1,newinst
NODENUM=0
CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

示例 5

TESTDB 数据库在数据库分区服务器 10 和 20 上有两个数据库分区。在这两个数据库分区服务器上，实例为 servinst，数据库路径为 /home/servinst。在这两个数据库分区服务器上，正在将数据库的名称更改为 SERVDB，并且正在将数据库路径更改为 /databases。另外，数据库分区服务器 20 上的日志目录正在从 /testdb_logdir 更改为 /servdb_logdir。

由于正在对这两个数据库分区进行更改，因此，必须为每个数据库分区创建一个配置文件，并且必须对具有相应配置文件的每个数据库分区服务器运行 db2relocatedb。

在数据库分区服务器 10 上，将使用以下配置文件:

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=10
```

在数据库分区服务器 20 上，将使用以下配置文件:

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=20
LOG_DIR=/testdb_logdir,/servdb_logdir
```

示例 6

MAINDB 数据库存在于 /home/maininst 路径上的 maininst 实例中。需要按如下所示更改四个表空间容器的位置:

```
需要将 /maininst_files/allconts/C0 移至 /MAINDB/C0
需要将 /maininst_files/allconts/C1 移至 /MAINDB/C1
需要将 /maininst_files/allconts/C2 移至 /MAINDB/C2
需要将 /maininst_files/allconts/C3 移至 /MAINDB/C3
```

在将物理目录和文件移至新位置后，可以将以下配置文件与 db2relocatedb 命令配合使用来更改数据库文件，以便它们识别新位置。

正在对所有容器执行类似的更改；即，正在将 /maininst_files/allconts/ 替换为 /MAINDB/，以便可以使用具有通配符的单个条目：

```
DB_NAME=MAINDB
DB_PATH=/home/maininst
INSTANCE=maininst
NODE_NUM=0
CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

使用说明

如果数据库所属的实例是变化的，那么在运行此命令前必须执行以下操作，以确保对实例和数据库支持文件进行更改：

- 如果要将数据库移至另一个实例，那么创建新的实例。
- 复制属于要复制到新实例所在系统的数据库的文件和设备。必须根据需要来更改路径名。但是，如果要将数据库文件移至的目录中已经有数据库，那么可以错误地覆盖现有 sqldbdir 文件，从而除去对现有数据库的引用。在这种情况下，无法使用 db2relocatedb 实用程序。除了 db2relocatedb 之外，另一种方法是“重定向复原”操作。
- 更改已复制的文件/设备的许可权，以便它们归实例所有者所有。

如果要更改实例，那么必须由新的实例所有者来运行该工具。

在分区数据库环境中，必须对需要更改的每个数据库分区运行此工具。必须为每个数据库分区提供独立的配置文件，该配置文件包含所更改的数据库分区的 NODENUM 值。例如，如果要更改数据库的名称，这将影响每个数据库分区，并且必须在每个数据库分区上都有独立配置文件的情况下运行 db2relocatedb 命令。如果要移动属于单一数据库分区的容器，那么只需要在该数据库分区上运行一次 db2relocatedb 命令。

不能使用 db2relocatedb 命令来重定位正在执行装入或者等待 LOAD RESTART 或 LOAD TERMINATE 命令完成的数据库。

局限性： 在分区数据库环境中，如果该节点是位于同一设备上的两个或多个逻辑分区之一，那么不能重定位整个节点。

db2look - DB2 统计信息和 DDL 抽取工具

抽取必需的数据定义语言（DDL）语句以便在测试数据库上重新生成生产数据库的数据库对象。db2look 命令根据对象类型生成 DDL 语句。

此工具可生成必需的 UPDATE 语句，用于在测试数据库中复制对象的统计信息。另外，它还可以用于生成 UPDATE DATABASE CONFIGURATION、UPDATE DATABASE MANAGER CONFIGURATION 命令以及 db2set 命令，以便测试数据库上与查询优化器相关的配置参数和注册表变量与生产数据库上的对应配置参数和注册表变量相匹配。

通常，具有一个包含生产系统的一部分数据的测试系统是很有利的。但是，为此类测试系统选择的存取方案并不一定与为生产系统选择的存取方案相同。必须更新测试系统的目录统计信息和配置参数以与生产系统上的那些目录统计信息和配置参数相匹配。在存取方案与生产系统上使用的那些存取方案相似的情况下，使用此工具可以创建一个测试数据库。

应该检查由 db2look 命令生成的 DDL 语句，这是因为这些语句可能无法准确重新生成原始 SQL 对象的所有特征。对于分区数据库环境上的表空间，如果某些数据库分区处于不活动状态，那么 DDL 可能不完整。请确保使用 ACTIVATE 命令激活所有数据库分区。

权限

对系统目录表的 SELECT 特权。

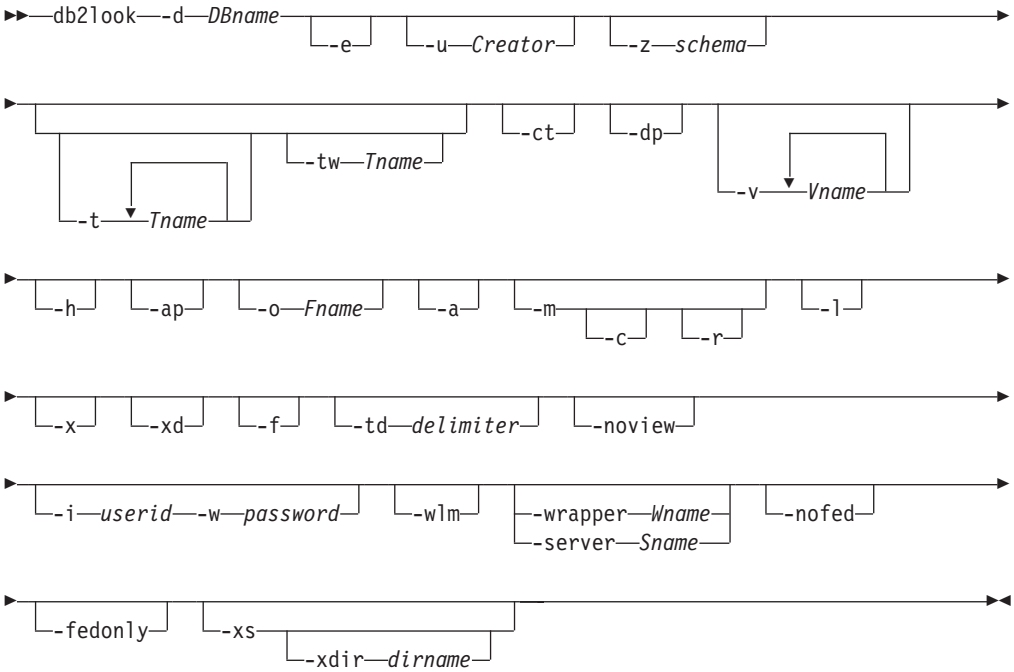
在某些情况下，例如，生成表空间容器 DDL（它调用 sqlbotcq、sqlbftcq 和 sqlbctcq API）时，您将需要下列其中一项特权:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

必需的连接

无

命令语法



命令参数

-d *DBname*

要查询的生产数据库的别名。*DBname* 可以是 DB2 数据库 Linux 版、UNIX 版和 Windows 版 或 DB2 版本 9.1 z/OS 版（DB2 z/OS 版）数据库的名称。如果 *DBname* 是 DB2 z/OS 版 数据库，那么 db2look 实用程序将抽取 OS/390 及 z/OS 对象的 DDL 和 UPDATE 统计信息语句。这些 DDL 和 UPDATE 统计信息语句适用于 DB2 数据库 Linux 版、UNIX 版和 Windows 版 数据库，但不适用于 DB2 z/OS 版 数据库。这对于想要抽取 OS/390 和 z/OS 对象并在 DB2 数据库 Linux 版、UNIX 版和 Windows 版数据库中重新创建这些对象的用户很有用。

如果 *DBname* 是 DB2 z/OS 版数据库，那么 db2look 命令的输出仅限于以下各项:

- 为表、索引、视图和用户定义的单值类型生成 DDL
- 为表、列、列分发和索引生成 UPDATE 统计信息语句

-e 抽取数据库对象的 DDL 语句。使用 -e 选项时抽取以下数据库对象的 DDL:

- 审计策略
- 模式
- 表
- 视图
- 具体化查询表（MQT）
- 别名
- 索引
- 触发器
- 序列
- 用户定义的单值类型
- 主键、引用完整性和检查约束
- 用户定义的结构化类型
- 用户定义的函数
- 用户定义的方法
- 用户定义的变换
- 包装器
- 服务器
- 用户映射
- 昵称
- 类型映射
- 函数模板
- 函数映射
- 索引规范
- 存储过程
- 角色
- 可信上下文

- 全局变量
- 安全标号组件
- 安全策略
- 安全标号

由 db2look 命令生成的 DDL 可用于成功地重新创建用户定义的函数。但是，某个用户定义的函数所引用的用户源代码（例如，EXTERNAL NAME 子句）必须可用，才能使用户定义的函数可用。

-u *Creator*

创建者标识。将输出限制于具有此创建者标识的对象。如果指定了选项 -a，那么将忽略此参数。该输出将不包括任何不可操作的对象。要显示不可操作的对象，请使用 -a 选项。

-z *schema*

模式名称。将输出限制于具有此模式名称的对象。该输出将不包括任何不可操作的对象。要显示不可操作的对象，请使用 -a 选项。如果未指定此参数，那么将抽取具有所有模式名称的对象。如果指定了 -a 选项，那么将忽略此参数。对于联合 DDL，将忽略此选项。

-t *Tname1 Tname2 ... TnameN*

表名列表。将输出限制于表列表中的特定表。最大表数是 30。表名之间使用空格进行分隔。区分大小写的名称和双字节字符集（DBCS）必须使用反斜杠和双引号定界符括起来，例如 \″ MyTabLe \″。对于多单词表名，必须使用引号将定界符括起来（例如，″\″My Table\″″），以防止命令行处理器将引号对按单个单词进行计算。如果多单词表名未使用反斜杠和双引号定界符括起来（例如，″My Table″），那么所有单词都将转换为大写并且 db2look 命令将查找单词全为大写的表（例如 ″MY TABLE″）。将 -t 与 -l 配合使用时，此组合支持 DB2 版本 9.5 中的分区表。

-tw *Tname*

为与 *Tname* 指定的模式条件相匹配的表名生成 DDL。另外，还为所有已返回表的所有从属对象生成 DDL。*Tname* 只能是单个值。*Tname* 中的下划线字符（_）可表示任意单个字符。百分号（%）表示一个包含零个或零个以上字符的字符串。*Tname* 中的任何其他字符仅表示其自身。指定 -tw 时，将忽略 -t 选项。

-ct 根据对象创建时间生成 DDL。根据对象创建时间生成 DDL 将不保证所有对象 DDL 均将以正确的依赖性顺序显示。如果同时指定了 -ct 选项，那么 db2look 命令仅支持以下选项：-e、-a、-u、-z、-t、-tw、-v、-l、-noview 和 -wlm。

-dp 在 CREATE 语句之前生成 DROP 语句。如果某一对象依赖于所删除的对象，那么 DROP 语句可能无效。例如，如果表依赖于某一模式，那么无法删除该模式，或者如果有任何其他类型、函数、触发器或表依赖于用户定义的类型/函数，那么无法删除该类型/函数。对于类型表，将仅为根表生成 DROP TABLE HIERARCHY 语句。删除表时将始终删除索引键、主键和外键以及约束，因此将不为它们生成 DROP 语句。如果表具有 RESTRICT ON DROP 属性，那么无法将该表删除。

-v *Vname1 Vname2 ... VnameN*

为指定视图生成 DDL。最大视图数是 30。如果指定了 -t 选项，那么将忽略 -v 选项。用于控制区分大小写的名称、DBCS 名称和多单词表名的规则同样适用于视图名称。

-h 显示帮助信息。当指定了此选项时，其他所有的选项都会被忽略，且只显示帮助信息。

-ap 生成必需的 AUDIT USING 语句以便将审计策略与其他数据库对象相关联。

-o *Fname*

将输出写入 *filename*.sql。如果未指定此选项，那么将输出写入标准输出。如果指定了一个带扩展名的文件名，那么会将输出写入该文件。

-a 指定此选项时，不会将输出限制于在特定创建者标识下创建的对象。将考虑由所有用户创建的所有对象，包括必需对象。例如，如果将此选项与 -e 选项一起指定，那么将为数据库中的所有对象抽取 DDL 语句。如果将此选项与 -m 选项一起指定，那么将为数据库中所有用户创建的表和索引抽取 UPDATE 统计信息语句。如果既不指定 -u 也不指定 -a，那么将使用环境变量 USER。在 UNIX 操作系统上，不必显式设置此变量；但是，在 Windows 系统上，USER 环境变量没有缺省值：必须在 SYSTEM 变量中设置 USER 变量，或者必须对会话发出 set USER=*username*。

-m 生成必需的 UPDATE 语句以复制有关表、统计信息视图、列和索引的统计信息。

-c 将此选项与 -m 选项一起指定时，db2look 命令不生成 COMMIT、CONNECT 和 CONNECT RESET 语句。缺省操作是生成这些语句。

-r 将此选项与 -m 选项一起指定时，db2look 命令不生成 RUNSTATS 命令。缺省操作是生成 RUNSTATS 命令。

-l 如果指定了此选项，那么 db2look 命令将为用户定义的表空间、数据库分区组和缓冲池生成 DDL。使用 -l 选项时抽取以下数据库对象的 DDL:

- 用户定义的表空间
- 用户定义的数据库分区组
- 用户定义的缓冲池

-x 如果指定了此选项，那么 db2look 命令将生成权限 DDL（例如 GRANT 语句）。

受支持的权限包括:

- 表：
  ALTER、SELECT、INSERT、DELETE、UPDATE、INDEX、REFERENCE、CONTROL
- 视图：SELECT、INSERT、DELETE、UPDATE、CONTROL
- 索引：CONTROL
- 模式：CREATEIN、DROPIN、ALTERIN
- 数据库：
  CREATEDB、BINDADD、CONNECT、CREWATE_NOT_FENCED、IMPLICIT_SCHEMA
- 用户定义的函数（UDF）：EXECUTE
- 用户定义的方法：EXECUTE
- 存储过程：EXECUTE
- 程序包：CONTROL、BIND、EXECUTE

- 列: UPDATE、REFERENCES
- 表空间: USE
- 序列: USAGE、ALTER
- 工作负载: USAGE
- 全局变量
- 角色
- 安全标号
- 免除权

-xd 如果指定了此选项，那么 db2look 命令将生成所有权限 DDL，其中包括创建对象时由 SYSIBM 向其授予权限的对象的权限 DDL。

-f 使用此选项来抽取那些影响查询优化器的配置参数和注册表变量。

-td *delimiter*
为使用 db2look 命令生成的 SQL 语句指定语句定界符。如果未指定此选项，那么缺省值为分号（;）。建议在指定 -e 选项时使用此选项。在这种情况下，抽取的对象可包含触发器或 SQL 例程。

-noview
如果指定了此选项，那么将不抽取 CREATE VIEW DDL 语句。

-i *userid*
处理远程数据库时使用此选项。

-w *password*
此参数与 -i 选项一起使用。此参数允许用户对位于远程系统上的数据库运行 db2look 命令。db2look 命令使用用户标识和密码来登录远程系统。如果处理的是远程数据库，那么远程数据库的版本必须与本地数据库的版本相同。db2look 命令没有下层或上层支持。

-wlm 此选项生成特定于 WLM 的 DDL 输出，该输出可用于为以下各项生成 CREATE 和 ALTER 语句:

- 直方图
- WLM 事件监视器
- 服务类
- 工作负载
- 阈值
- 工作类集
- 工作操作集

-wrapper *Wname*
为应用于此包装器的联合对象生成 DDL 语句。可能生成的联合 DDL 语句包括: CREATE WRAPPER、CREATE SERVER、CREATE USER MAPPING、CREATE NICKNAME、CREATE TYPE MAPPING、CREATE FUNCTION ... AS TEMPLATE、CREATE FUNCTION MAPPING、CREATE INDEX SPECIFICATION 和 GRANT（GRANT 语句向别名、服务器和索引授予特权）。仅支持一个包装器名称; 如果指定了零个或多个包装器名称，那么将返回错误。此选项不支持非关系数据源。

-server *Sname*

为应用于此服务器的联合对象生成 DDL 语句。可能生成的联合 DDL 语句包括：CREATE WRAPPER、CREATE SERVER、CREATE USER MAPPING、CREATE NICKNAME、CREATE TYPE MAPPING、CREATE FUNCTION ... AS TEMPLATE、CREATE FUNCTION MAPPING、CREATE INDEX SPECIFICATION 和 GRANT（GRANT 语句向别名、服务器和索引授予特权）。仅支持一个服务器名称；如果指定了零个或多个服务器名称，那么将返回错误。此选项不支持非关系数据源。

-nofed

指定将不生成任何联合 DDL 语句。指定此选项时，将忽略 -wrapper 和 -server 选项。

-fedonly

指定将仅生成联合 DDL 语句。

-xs 导出在目标数据库中注册 XML 模式和 DTD 时所必需的所有文件，并生成适当的命令以注册这些 XML 模式和 DTD。由 -u、-z 和 -a 选项控制将导出的 XSR 对象集。

-xdir *dirname*

将所导出的 XML 相关文件放入给定路径中。如果未指定此选项，那么将所有 XML 相关文件导出到当前目录。

示例

- 为对象（由用户 walid 在数据库 DEPARTMENT 中创建）生成 DDL 语句。db2look 输出将发送至文件 db2look.sql：

```
db2look -d department -u walid -e -o db2look.sql
```

- 为对象（由用户 walid 在数据库 DEPARTMENT 中创建，具有模式名称 ianhe）生成 DDL 语句。db2look 输出将发送至文件 db2look.sql：

```
db2look -d department -u walid -z ianhe -e -o db2look.sql
```

- 生成 UPDATE 语句以复制数据库对象（由用户 walid 在数据库 DEPARTMENT 中创建）的统计信息。输出将发送至文件 db2look.sql：

```
db2look -d department -u walid -m -o db2look.sql
```

- 为对象（由用户 walid 创建）生成 DDL 语句以及生成 UPDATE 语句以复制有关数据库对象（由同一用户创建）的统计信息。db2look 输出将发送至文件 db2look.sql：

```
db2look -d department -u walid -e -m -o db2look.sql
```

- 为对象（由所有用户在数据库 DEPARTMENT 中创建）生成 DDL 语句。db2look 输出将发送至文件 db2look.sql：

```
db2look -d department -a -e -o db2look.sql
```

- 为所有用户定义的数据库分区组、缓冲池和表空间生成 DDL 语句。db2look 输出将发送至文件 db2look.sql：

```
db2look -d department -l -o db2look.sql
```

- 为优化器相关的数据库和数据库管理器配置参数生成 UPDATE 语句，以及为数据库 DEPARTMENT 中的优化器相关注册表变量生成 db2set 语句。db2look 输出将发送至文件 db2look.sql：

```
db2look -d department -f -o db2look.sql
```

- 为数据库 DEPARTMENT 中的所有对象生成 DDL、生成 UPDATE 语句以复制数据库 DEPARTMENT 中有关所有表和索引的统计信息、为优化器相关数据库和数据库管理器配置参数生成 GRANT 权限语句和 UPDATE 语句、为优化器相关注册表变量生成 db2set 语句，以及为数据库 DEPARTMENT 中的所有用户定义的数据库分区组、缓冲池和表空间生成 DDL。输出将发送至文件 db2look.sql:

  ```
  db2look -d department -a -e -m -l -x -f -o db2look.sql
  ```

- 为数据库 DEPARTMENT 中的所有对象（包括由原始创建者创建的对象）生成所有权限 DDL 语句。（在这种情况下，由 SYSIBM 在创建对象时授予这些权限。）db2look 输出将发送至文件 db2look.sql:

  ```
  db2look -d department -xd -o db2look.sql
  ```

- 为对象（由所有用户在数据库 DEPARTMENT 中创建）生成 DDL 语句。db2look 输出将发送至文件 db2look.sql:

  ```
  db2look -d department -a -e -td % -o db2look.sql
  ```

  输出随后可供 CLP 读取:

  ```
  db2 -td% -f db2look.sql
  ```

- 为数据库 DEPARTMENT 中的对象创建 DDL 语句，不包括 CREATE VIEW 语句。db2look 输出将发送至文件 db2look.sql:

  ```
  db2look -d department -e -noview -o db2look.sql
  ```

- 为数据库 DEPARTMENT 中与指定表相关的对象生成 DDL 语句。db2look 输出将发送至文件 db2look.sql:

  ```
  db2look -d department -e -t tab1 \"My TaBlE2\" -o db2look.sql
  ```

- 为联合数据库 FEDDEPART 中的所有联合和非联合对象生成 DDL 语句。对于联合 DDL 语句，仅生成那些适用于指定包装器 FEDWRAP 的联合 DDL 语句。db2look 输出将发送至标准输出:

  ```
  db2look -d feddepart -e -wrapper fedwrap
  ```

- 生成一个仅包含非联合 DDL 语句的脚本文件。可对联合数据库（FEDDEPART）运行以下系统命令，但仅生成类似于对未联合的数据库运行此命令时所生成的输出。db2look 输出将发送至文件 out.sql:

  ```
  db2look -d feddepart -e -nofed -o out
  ```

- 为在数据库 DEPARTMENT 中具有模式名称 walid 的对象生成 DDL 语句。注册任何附带的 XML 模式和 DTD 时需要的文件将导出到当前目录。db2look 输出将发送至文件 db2look.sql:

  ```
  db2look -d department -z walid -e -xs -o db2look.sql
  ```

- 为对象（由所有用户在数据库 DEPARTMENT 中创建）生成 DDL 语句。注册任何附带的 XML 模式和 DTD 时需要的文件将导出到目录 /home/ofer/ofer/。db2look 输出将发送至标准输出:

  ```
  db2look -d department -a -e -xs -xdir /home/ofer/ofer/
  ```

- 以互斥方式在数据库 DEPARTMENT 中生成特定于 WLM 的 DDL。

  ```
  db2look -d department -wlm
  ```

  为数据库 DEPARTMENT 中的所有对象生成 DDL。

  ```
  db2look -d department -wlm -e -l
  ```

使用说明

在 Windows 操作系统上，必须从 DB2 命令窗口中运行 db2look 命令。

现有的若干个选项可支持联合环境。在联合环境中使用以下 db2look 命令行选项:

- -ap

  如果使用此选项，那么将生成 AUDIT USING 语句。

- -e

  如果使用此选项，那么将生成联合 DDL 语句。

- -x

  如果使用此选项，那么将生成 GRANT 语句以向联合对象授予特权。

- -xd

  如果使用此选项，那么将生成联合 DDL 语句以向联合对象添加系统授予的特权。

- -f

  如果使用此选项，那么将从数据库管理器配置中抽取与联合相关的信息。

- -m

  如果使用此选项，那么将抽取昵称的统计信息。

- -wlm

  如果使用此选项，那么将输出特定于 WLM 的 DDL。

需要在数据库管理器配置中启用使用联合系统的能力，以便创建联合 DDL 语句。在 db2look 命令生成脚本文件后，您必须将**联合**配置参数设置为 YES，然后再运行该脚本。

需要修改输出脚本以便为 CREATE USER MAPPING 语句添加远程密码。

需要通过向那些用于将 DB2 系列实例定义为数据源的 CREATE SERVER 语句添加 AUTHORIZATION 和 PASSWORD，来修改 db2look 命令输出脚本。

-tw 选项的用法如下所示:

- 要为 DEPARTMENT 数据库中与名称以 abc 开头的表相关联的对象生成 DDL 语句，并将输出发送至 db2look.sql 文件:

  ```
  db2look -d department -e -tw abc% -o db2look.sql
  ```

- 要为 DEPARTMENT 数据库中与使用 d 作为名称的第二个字符的表相关联的对象生成 DDL 语句，并将输出发送至 db2look.sql 文件:

  ```
  db2look -d department -e -tw _d% -o db2look.sql
  ```

- 评估哪些表名与 *Tname* 自变量所指定的模式相匹配时，db2look 命令将使用 LIKE 谓词。因为使用了 LIKE 谓词，所以如果 _ 字符或 % 字符是表名的一部分，那么必须紧接 _ 或 % 之前使用反斜杠 (\) 转义字符。在这种情况下，不能在 *Tname* 中使用 _ 或 % 作为通配符。例如，要为 DEPARTMENT 数据库中与名称的第一个字符和最后一个字符均未使用百分号的表相关联的对象生成 DDL 语句:

  ```
  db2look -d department -e -tw string\%string
  ```

- 必须使用反斜杠和双引号将区分大小写的名称、DBCS 名称以及多单词表名和视图名称括起来。例如:

  ```
  \"My TabLe\"
  ```

  如果未使用反斜杠和双引号定界符将多字节字符集（MBCS）或双字节字符集（DBCS）名称括起来，并且该名称包含与小写字符相同的字节，那么会将该名称转换为大写并且 db2look 将查找具有所转换名称的数据库对象。因此，将不抽取 DDL 语句。
- -tw 选项可以与 -x 选项配合使用（以生成 GRANT 特权）、与 -m 选项配合使用（以返回表和列统计信息）以及与 -l 选项配合使用（以便为用户定义的表空间、数据库分区组和缓冲池生成 DDL）。如果将 -t 选项与 -tw 选项一起指定，那么将忽略 -t 选项（及其关联的 *Tname* 自变量）。
- -tw 选项不能用于为位于联合数据源、DB2 通用数据库 z/OS 版和OS/390 版、 DB2 for i5/OS® 或 DB2 服务器 VSE & VM 版 上的表（及其关联的对象）生成 DDL。
- 仅通过 CLP 支持 -tw 选项。

在使用数据库分区功能的系统上请求 DDL 时，将显示一条警告消息，而不是显示存在于不活动数据库分区上的表空间的 DDL。必须激活所有数据库分区，才能确保为所有表空间生成正确的 DDL。

抽取类型为数组的安全标号组件的 DDL 时，所抽取的 DDL 可能无法生成这样一个安全标号组件：其内部表示（例如，该数组中元素的编码）正好与数据库（从中抽取 db2look）中安全标号组件的内部表示相匹配。改变一个类型为数组的安全标号组件并向其添加一个或多个元素后，会发生这种情况。在此类情况下，从一个表中抽取并移入另一个表的数据以及根据 db2look 输出创建的数据将没有对应的安全标号值，因此会使对新表的保护遭到破坏。

相关信息

昵称列和索引名

更改迁移应用程序

第 6 章 文件格式和数据类型

Export/Import/load 实用程序文件格式

以下描述了 DB2 export 实用程序、import 实用程序和 load 实用程序支持的五种操作系统文件格式:

DEL 定界 ASCII，用于各种数据库管理器和文件管理器之间的数据交换。这是用来存储数据的常用方法，它使用特殊字符定界符来隔开列值。

ASC 非定界 ASCII，用于从其他应用程序导入或装入数据，这些应用程序使用已对齐的列数据创建纯文本文件。

PC/IXF

PC 版本的集成交换格式（IXF），这是在数据库管理器中进行数据交换的首选格式。PC/IXF 是对数据库表的结构化描述，该数据库表包含内部表的外部表示。

WSF 工作表格式，用于 Lotus 1-2-3 和 Symphony 之类的产品的数据交换。load 实用程序不支持此文件格式。

CURSOR

对 SQL 查询声明的游标。仅 load 实用程序支持此文件类型。

使用 DEL、WSF 或 ASC 数据文件格式时，应在导入文件前定义表，包括其列名和数据类型。操作系统文件字段中的数据类型将转换为数据库表中相应的数据类型。import 实用程序接受带有轻微不兼容问题的数据，包括导入时可能带有填充或截断的字符数据，以及导入到不同类型的数字字段中的数字数据。

在使用 PC/IXF 数据文件格式时，开始导入操作之前表不一定要存在。但不必定义用户定义的单值类型（UDT），否则会接收到未定义名称错误（SQL0204N）。同样，导出至 PC/IXF 数据文件格式时，UDT 将存储在输出文件中。

使用 CURSOR 文件类型时，必须在开始装入操作之前定义该表，包括其列名和数据类型。SQL 查询的列类型必须与目标表中的相应列类型兼容。在开始装入操作之前，指定的游标不一定要打开。无论是否使用游标来访存行，load 实用程序都将处理与指定游标相关联的查询的完整结果。

在平台之间移动数据 - 文件格式注意事项

在跨平台导出、导入或装入数据时，兼容性至关重要。下列各节描述在不同操作系统之间移动数据时的 PC/IXF、定界 ASCII（DEL）和 WSF 文件格式注意事项。

PC/IXF 文件格式

跨平台传输数据时，建议您使用 PC/IXF 文件格式。PC/IXF 文件使 load 实用程序或 import 实用程序能够以独立于机器的方式来处理（通常依赖于机器的）数字数据。例如，数字数据在 Intel® 与其他硬件体系结构上的存储和处理方式是不同的。

为了在 DB2 系列中的所有产品间提供 PC/IXF 文件的兼容性，export 实用程序使用 Intel 格式的数字数据创建文件，而 import 实用程序将使用此格式。

根据硬件平台，DB2 产品会在导出和导入操作期间 Intel 与非 Intel 格式之间转换数字值（使用字节逆转）。

基于 UNIX 的 DB2 数据库实现不会在导出期间创建多部件 PC/IXF 文件。但是，它们允许您导入由 DB2 创建的多部件 PC/IXF 文件。导入此类型的文件时，所有部件应该在同一个目录中，否则会返回错误。

由基于 UNIX 的 DB2 export 实用程序实现创建的单部件 PC/IXF 文件可由 DB2 数据库 Windows 版导入。

定界 ASCII（DEL）文件格式

根据创建 DEL 文件的操作系统不同，DEL 文件是有差别的。这些差别是:

- 行分隔符
  - 基于 UNIX 的文本文件使用换行（LF）字符。
  - 不基于 UNIX 的文本文件使用回车符/换行符（CRLF）序列。
- 文件结束符
  - 基于 UNIX 的文本文件没有文件结束符。
  - 不基于 UNIX 的文本文件有文件结束符（X'1A'）。

由于 DEL 导出文件是文本文件，所以可以将它们从一个操作系统传输到另一个操作系统。如果以文本方式传输文件，文件传输程序就可以处理依赖于操作系统的差别；以二进制方式传输文件时，不会执行行分隔符和文件结束符转换操作。

注: 如果字符数据字段包含行分隔符，在文件传输期间就会转换这些字符。此项转换将导致意外地更改数据，因此，建议您不要使用 DEL 导出文件来跨平台移动数据。而是，应该使用 PC/IXF 文件格式。

WSF 文件格式

WSF 格式文件中的数字数据是使用 Intel 机器格式存储的。此格式允许在不同 Lotus 操作环境（如基于 Intel 的系统和基于 UNIX 的系统）之间传输和使用 Lotus WSF 文件。

由于内部格式一致，所以在另一平台上运行的 Lotus 1-2-3 或 Symphony 可以使用从 DB2 产品中导出的 WSF 文件。DB2 产品还可导入在不同平台上创建的 WSF 文件。

在操作系统之间传输 WSF 文件时，应使用二进制文本（而非文本方式）。

注: 不要使用 WSF 文件格式来在不同平台上的 DB2 数据库之间传输数据，这样做可能会导致丢失数据。而是，应该使用 PC/IXF 文件格式。

定界 ASCII（DEL）文件格式

定界 ASCII（DEL）文件是带有行定界符和列定界符的顺序 ASCII 文件。每个 DEL 文件都是一个 ASCII 字符流，该字符流由先按行排序然后按列排序的单元值组成。数据流中的各行由行定界符分隔；在每一行中，各个单元值由列定界符分隔。

下表描述可导入或可因为导出操作而生成的 DEL 文件的格式。

```
DEL file ::= Row 1 data || Row delimiter ||
             Row 2 data || Row delimiter ||
             .
             .
             .
             Row n data || Optional row delimiter

Row i data ::= Cell value(i,1) || Column delimiter ||
               Cell value(i,2) || Column delimiter ||
               .
               .
               .
               Cell value(i,m)

Row delimiter ::= ASCII line feed sequence[a]

Column delimiter ::= Default value ASCII comma (,)[b]

Cell value(i,j) ::= Leading spaces
                 || ASCII representation of a numeric value
                    (integer, decimal, or float)
                 || Delimited character string
                 || Non-delimited character string
                 || Trailing spaces

Non-delimited character string ::= A set of any characters except a
                                   row delimiter or a column delimiter

Delimited character string ::= A character string delimiter ||
                               An extended character string ||
                               A character string delimiter ||
                               Trailing garbage

Trailing garbage ::= A set of any characters except a row delimiter
                     or a column delimiter

Character string delimiter ::= Default value ASCII double quotation
                               marks (")[c]

extended character string ::= || A set of any characters except a
                                 row delimiter or a character string
                                 delimiter if the NODOUBLEDEL
                                 modifier is specified
                              || A set of any characters except a
                                 row delimiter or a character string
                                 delimiter if the character string
                                 is not part of two consecutive
                                 character string delimiters
                              || A set of any characters except a
                                 character string delimiter if the
                                 character string delimiter is not
                                 part of two consecutive character
                                 string delimiters, and the DELPRIORITYCHAR
                                 modifier is specified

End-of-file character ::= Hex '1A' (Windows operating system only)

ASCII representation of a numeric value[d] ::= Optional sign '+' or '-'
    || 1 to 31 decimal digits with an optional decimal point before,
       after, or between two digits
    || Optional exponent

Exponent ::= Character 'E' or 'e'
    || Optional sign '+' or '-'
    || 1 to 3 decimal digits with no decimal point
```

```
Decimal digit ::= Any one of the characters '0', '1', ... '9'

Decimal point ::= Default value ASCII period (.)[e]
```

- [a] 假定记录定界符是换行符，即 ASCII x0A。在 Windows 操作系统上生成的数据可以使用双字节的回车符/换行符标准，即 0x0D0A。EBCDIC 代码页的数据应该使用 EBCDIC LF 字符（0x25）作为记录定界符（可使用 LOAD 命令的 codepage 文件类型修饰符装入 EBCDIC 数据）。
- [b] 可使用 coldel 文件类型修饰符指定列定界符。
- [c] 可使用 chardel 文件类型修饰符指定字符串定界符。

  注: 定界符的缺省优先级为:

  1. 记录定界符
  2. 字符定界符
  3. 列定界符
- [d] 如果数字值的 ASCII 表示包含指数，那么它是 FLOAT 常量。如果它有小数点而没有指数，那么它是 DECIMAL 常量。如果它没有小数点和指数，那么它是 INTEGER 常量。
- [e] 可使用 decpt 文件类型修饰符指定小数点字符。

export 实用程序会将嵌入在列数据中的每个字符串定界符字节（缺省值为双引号或 x22）替换为两个字符串定界符字节（即，长度加倍）。这样做是为了让导入语法分析例程能够区分用于定义列开头或结尾的字符串定界符字节与嵌入在列数据中的字符串定界符字节。将导出的 DEL 文件用于除 export 实用程序外的某个应用程序时请务必小心，并且还要注意长度加倍的字符串定界符会出现在“FOR BIT”二进制列数据中。

DEL 数据类型描述

表 50. DEL 文件格式可接受的数据类型格式

数据类型	export 实用程序创建的文件中的格式	import 实用程序可接受的格式
BIGINT	范围在 -9 223 372 036 854 775 808 到 9 223 372 036 854 775 807 之间的 INTEGER 常量。	范围在 -9 223 372 036 854 775 808 到 9 223 372 036 854 775 807 之间的数字值的 ASCII 表示。十进制数字和浮点型数字被截断为整数值。
BLOB 和 CLOB	用双引号之类的字符定界符括起来的字符数据。	定界或非定界字符串。该字符串用作数据库列值。
BLOB_FILE 和 CLOB_FILE	每个 BLOB/CLOB 列的字符数据存储在个别文件中，文件名用字符定界符括起来。	包含数据的文件的定界或非定界名称。
CHAR	用双引号之类的字符定界符括起来的字符数据。	定界或非定界字符串。必要时，将截断字符串或用空格填充字符串（X’20’）以与数据库列的宽度相匹配。

表 50. DEL 文件格式可接受的数据类型格式 (续)

数据类型	export 实用程序创建的文件中的格式	import 实用程序可接受的格式
DATE	*yyyymmdd*（年月日），无字符定界符。例如: 19931029 或者，可使用 DATESISO 选项来指定以 ISO 格式导出所有日期值。	定界或非定界字符串，该字符串包含与目标数据库的地域代码一致的 ISO 格式的日期值，或者是格式为 *yyyymmdd* 的非定界字符串。
DBCLOB（仅适用于 DBCS）	图形数据将作为定界字符串导出。	定界或非定界字符串，长度为偶数字节。该字符串用作数据库列值。
DBCLOB_FILE（仅适用于 DBCS）	每个 DBCLOB 列的字符数据存储在个别文件中，文件名用字符定界符括起来。	包含数据的文件的定界或非定界名称。
DB2SECURITYLABEL	列数据将作为用引号（“）引起来的“原始”数据导出。在 SELECT 语句中使用 SECLABEL_TO_CHAR 标量函数以将值转换为安全标号字符串格式。	缺省情况下，将数据文件中的值视为组成该安全标号的内部表示的实际字节数。假定用引号（″ ″）对该值进行定界。
DECIMAL	带有要导出的字段的精度和标度的 DECIMAL 常量。可使用 `decplusblank` 文件类型修饰符来指定: 正十进制值的前面将加上空格而不是加号（+）。	数字值的 ASCII 表示，该值未超过在其中导入字段的数据库列的范围。如果输入值的小数点以后的位数超过数据库列可以容纳的位数，那么超过的位数将被截断。
FLOAT(long)	范围在 -10E307 到 10E307 之间的 FLOAT 常量。	范围在 -10E307 到 10E307 之间的数字值的 ASCII 表示。
GRAPHIC（仅适用于 DBCS）	图形数据将作为定界字符串导出。	定界或非定界字符串，长度为偶数字节。必要时，将截断字符串或用双字节空格填充字符串（如 X’8140’）以与数据库列的宽度相匹配。
INTEGER	范围在 -2 147 483 648 到 2 147 483 647 之间的 INTEGER 常量。	范围在 -2 147 483 648 到 2 147 483 647 之间的数字值的 ASCII 表示。十进制数字和浮点型数字被截断为整数值。
LONG VARCHAR	用双引号之类的字符定界符括起来的字符数据。	定界或非定界字符串。该字符串用作数据库列值。
LONG VARGRAPHIC（仅适用于 DBCS）	图形数据将作为定界字符串导出。	定界或非定界字符串，长度为偶数字节。该字符串用作数据库列值。
SMALLINT	范围在 -32 768 到 32 767 之间的 INTEGER 常量。	范围在 -32 768 到 32 767 之间的数字值的 ASCII 表示。十进制数字和浮点型数字被截断为整数值。

表 50. DEL 文件格式可接受的数据类型格式 (续)

数据类型	export 实用程序创建的文件中的格式	import 实用程序可接受的格式
TIME	*hh.mm.ss*（小时分钟秒）。这是用字符定界符括起来的 ISO 格式的时间值。例如：“09.39.43”	定界或非定界字符串，该字符串包含与目标数据库的地域代码一致的格式的日期值。
TIMESTAMP	*yyyy-mm-dd-hh.mm.ss.nnnnnn*（年月日小时分钟秒微秒）。用字符定界符括起来的表示日期和时间的字符串。	定界或非定界字符串，该字符串包含可存储在数据库中的时间戳记值。
VARCHAR	用双引号之类的字符定界符括起来的字符数据。	定界或非定界字符串。必要时，将截断字符串以与数据库列的最大宽度相匹配。
VARGRAPHIC（仅适用于 DBCS）	图形数据将作为定界字符串导出。	定界或非定界字符串，长度为偶数字节。必要时，将截断字符串以与数据库列的最大宽度相匹配。

示例 DEL 文件

以下是一个 DEL 文件示例。每一行都以换行符序列结尾（在 Windows 操作系统上，每一行都以回车符/换行符序列结尾）。

```
"Smith, Bob",4973,15.46
"Jones, Bill",12345,16.34
"Williams, Sam",452,193.78
```

以下示例说明了非定界字符串的使用。由于字符数据包含逗号，所以已将列定界符更改为分号。

```
Smith, Bob;4973;15.46
Jones, Bill;12345;16.34
Williams, Sam;452;193.78
```

注:

1. 空格（X’20’）永远不能作为有效定界符。
2. 在导入期间将删除第一个字符前面的空格或者单元格值中最后一个字符后面的空格。不会删除单元格值中间嵌入的空格。
3. 由于句点（.）与时间戳记值中的句点冲突，所以它不是有效的字符字符串定界符。
4. 对于纯 DBCS（图形）、混合 DBCS 和 EUC 来说，定界符的范围是 x00 到 x3F。
5. 对于使用 EBCDIC 代码页指定的 DEL 数据来说，定界符可能与 shift-in 和 shift-out DBCS 字符不一致。
6. 在 Windows 操作系统上，字符定界符外部第一次出现的文件结束符（X’1A’）指示文件结束。不会导入任何后续数据。
7. 空值表示通常应该具有值的单元格缺少单元格值，也可以表示一串空格。

8. 由于某些产品将字符字段的长度限制为 254 或 255 个字节，所以，每当选择导出最大长度超过 254 个字节的字符列时，export 实用程序就会生成警告消息。import 实用程序可以接受与最长的 LONG VARCHAR 和 LONG VARGRAPHIC 列等长的字段。

移动数据时的定界符注意事项

移动定界 ASCII（DEL）文件时，一定要确保移动的数据不会因为定界字符识别问题而导致无意中发生改变。为帮助避免发生这些错误，DB2 会强制实施若干限制并提供了许多文件类型修饰符。

定界符限制

有许多限制可帮助避免所选定界字符被视为移动数据的一部分。首先，定界符是互斥的。其次，定界符不能是二进制零、换行符、回车符或空格。而且缺省小数点（.）不能是字符串定界符。最后，在 DBCS 环境中，不支持竖线（|）字符定界符。

ASCII 系列代码页和 EBCDIC 系列代码页对下列字符的指定是不同的:

- Shift-In 字符（0x0F）和 Shift-Out 字符（0x0E）不能是 EBCDIC MBCS 数据文件的定界符。
- MBCS、EUC 或 DBCS 代码页的定界符不能大于 0x40，但 EBCDIC MBCS 数据的缺省小数点（0x4b）除外。
- 采用 ASCII 代码页或 EBCDIC MBCS 代码页的数据文件的缺省定界符是:
  - 字符串定界符: "（0x22，双引号）
  - 列定界符: ,（0x2c，逗号）
- 采用 EBCDIC SBCS 代码页的数据文件的缺省定界符是:
  - 字符串定界符: "（0x7F，双引号）
  - 列定界符: ,（0x6B，逗号）
- ASCII 数据文件的缺省小数点是 0x2e（句点）。
- EBCDIC 数据文件的缺省小数点是 0x4B（句点）。
- 如果服务器的代码页与客户机的代码页不同，那么建议指定非缺省定界符的十六进制表示法。例如,

```
db2 load from ... modified by chardel0x0C coldelX1e ...
```

数据移动期间的定界符问题

双字符定界符

缺省情况下，对于 DEL 文件的基于字符的字段，字段中的任何字符定界符实例都用双字符定界符表示。例如，假定字符定界符是双引号，如果导出文本 `I am 6" tall.`，那么 DEL 文件中的输出文本显示为 `"I am 6"" tall."`相反，如果 DEL 文件中的输入文本为 `"What a ""nice"" day!"`，那么导入的文本为 `What a "nice" day!`

nodoubledel

可通过指定 `nodoubledel` 文件类型修饰符对 import、export 和 load 实用程序禁用双字符定界符行为。但要注意的是，双字符定界符行为是为了避免解析错误。对导出使用 `nodoubledel` 时，字符定界符出现在字符字段中时不会显示为双字符。对导入和装入使用 `nodoubledel` 时，双字符定界符不会解释为字符定界符的字面值实例。

nochardel
对导出使用 nochardel 文件类型修饰符时，字符字段不会用字符定界符括起来。对导入和装入使用 nochardel 时，字符定界符不会被视作特殊字符并且会解释为实际数据。

chardel
可使用其他文件类型修饰符，以通过手动方式避免缺省定界符与数据之间混淆。chardel 文件类型修饰符将单字符 x 指定为要使用的字符串定界符，以替代缺省情况下使用的双引号。

coldel
同样，如果要避免将缺省情况下使用的逗号用作列定界符，可使用 coldel，它会将单字符 x 指定为列数据定界符。

delprioritychar
移动 DEL 文件时另一个需要注意的问题就是保留定界符的正确优先顺序。定界符的缺省优先级为：行，字符，列。但是，某些应用程序依赖于以下优先级：字符，行，列。例如，如果使用缺省优先级，那么 DEL 数据文件：

```
"Vincent <row delimiter> is a manager",<row delimiter>
```

将解释为以下两行：Vincent 和 is a manager，原因是 <row delimiter> 优先于字符定界符（"）。如果使用 delprioritychar，那么字符定界符（"）优先于行定界符（<row delimiter>），这意味着同一 DEL 文件将正确地解释为一行：Vincent is a manager。

非定界 ASCII（ASC）文件格式

非定界 ASCII 格式（对于 Import 和 load 实用程序来说称为 ASC）有两种变体：定长 ASC 和变长 ASC。对于定长 ASC 来说，所有记录都是定长的。对于变长 ASC 来说，记录由行定界符（始终是换行符）定界。在非定界 ASCII 中，术语*非定界*表示列值未由定界符分隔。

在导入或装入 ASC 数据时，如果指定了 reclen 文件类型修饰符，那么表示数据文件是定长 ASC 文件。如果未指定该修饰符，那么表示该数据文件是变长 ASC 文件。

非定界 ASCII 格式可以用于与任何使用列数据格式的 ASCII 产品（包括字处理器）交换数据。每个 ASC 文件都是 ASCII 字符流，该字符流由按行列排序的数据值组成。数据流中的各行由行定界符分隔。在一行中，每一列都由开始结束位置对（由 IMPORT 参数指定）定义。每个位置对都表示一行中以字节位置形式指定的位置。一行内的第一个位置为字节位置 1。每个位置对的第一个元素是该列的开始字节，每个位置对的第二个元素是该列的结束字节。各个列可以重叠。ASC 文件中的每一行的列定义相同。

ASC 文件的定义方式如下所示：

```
ASC file ::= Row 1 data || Row delimiter ||
                           Row 2 data || Row delimiter ||
             .
             .
             .
                           Row n data

Row i data ::= ASCII characters || Row delimiter

Row Delimiter ::= ASCII line feed sequence[a]
```

- [a] 假定记录定界符是换行符，即 ASCII x0A。在 Windows 操作系统上生成的数据可以使用双字节的回车符/换行符标准，即 0x0D0A。EBCDIC 代码页的数据应该使用 EBCDIC LF 字符（0x25）作为记录定界符（可使用 LOAD 命令的 codepage 文件类型修饰符装入 EBCDIC 数据）。永远不会将记录定界符解释成属于数据字段。

ASC 数据类型描述

表 51. ASC 文件格式可接受的数据类型格式

数据类型	import 实用程序可接受的格式
BIGINT	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的常量。如果各个值不在范围 -9 223 372 036 854 775 808 到 9 223 372 036 854 775 807 之间，那么会拒绝这些值。十进制数字被截断为整数值。逗号、句点或冒号被视为小数点。不允许使用千位分隔符。 开始和结尾位置应指定宽度不超过 50 个字节的字段。整数、十进制数和浮点数的尾数不能超过 31 位。浮点数的指数不能超过 3 位。
BLOB/CLOB	一个字符串。必要时，将截断字符串的右边以与目标列的最大宽度相匹配。如果 ASC *截断空白*选项有效，那么将去掉原始字符串或被截断字符串的尾部空格。
BLOB_FILE、CLOB_FILE 和 DBCLOB_FILE（仅适用于 DBCS）	包含数据的文件的定界或非定界名称。
CHAR	一个字符串。必要时，将截断字符串的右边部分或用空格填充字符串的右边以与目标列的宽度相匹配。
DATE	表示日期值的字符串，该日期值的格式与目标数据库的地域代码一致。 开始和结尾位置应指定宽度在日期的外部表示的范围内的字段。
DBCLOB（仅适用于 DBCS）	偶数字节的字符串。奇数字节的字符串无效并且是不可接受的。必要时，将截断有效字符串的右边部分以与目标列的最大宽度相匹配。
DECIMAL	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的常量。如果各个值不在要导入至其中的数据库列的范围内，那么会拒绝这些值。如果输入值的小数点以后的位数超过数据库列的标度，那么超过的位数将被截断。逗号、句点或冒号被视为小数点。不允许使用千位分隔符。 开始和结尾位置应指定宽度不超过 50 个字节的字段。整数、十进制数和浮点数的尾数不能超过 31 位。浮点数的指数不能超过 3 位。
FLOAT(long)	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的常量。所有值都有效。逗号、句点或冒号被视为小数点。大写或小写 E 将作为 FLOAT 常量的指数的开头部分接受。 开始和结尾位置应指定宽度不超过 50 个字节的字段。整数、十进制数和浮点数的尾数不能超过 31 位。浮点数的指数不能超过 3 位。

表 51. ASC 文件格式可接受的数据类型格式 (续)

数据类型	import 实用程序可接受的格式
GRAPHIC（仅适用于 DBCS）	偶数字节的字符串。奇数字节的字符串无效并且是不可接受的。必要时，将截断有效字符串的右边部分或用双字节空格（0x8140）填充有效字符串的右边以与目标列的最大长度相匹配。
INTEGER	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的常量。如果各个值不在范围 -2 147 483 648 到 2 147 483 647 之间，那么会拒绝这些值。十进制数字被截断为整数值。逗号、句点或冒号被视为小数点。不允许使用千位分隔符。 开始和结尾位置应指定宽度不超过 50 个字节的字段。整数、十进制数和浮点数的尾数不能超过 31 位。浮点数的指数不能超过 3 位。
LONG VARCHAR	一个字符串。必要时，将截断字符串的右边以与目标列的最大宽度相匹配。如果 ASC *截断空白*选项有效，那么将去掉原始字符串或被截断字符串的尾部空格。
LONG VARGRAPHIC（仅适用于 DBCS）	偶数字节的字符串。奇数字节的字符串无效并且是不可接受的。必要时，将截断有效字符串的右边部分以与目标列的最大宽度相匹配。
SMALLINT	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的常量。如果各个值不在范围 -32 768 到 32 767 之间，那么会拒绝这些值。十进制数字被截断为整数值。逗号、句点或冒号被视为小数点。不允许使用千位分隔符。 开始和结尾位置应指定宽度不超过 50 个字节的字段。整数、十进制数和浮点数的尾数不能超过 31 位。浮点数的指数不能超过 3 位。
TIME	表示时间值的字符串，该时间值的格式与目标数据库的地域代码一致。 开始和结尾位置应指定宽度在时间的外部表示的范围内的字段。
TIMESTAMP	表示可存储在数据库中的时间戳记值的字符串。 开始和结尾位置应指定宽度在时间戳记的外部表示的范围内的字段。
VARCHAR	一个字符串。必要时，将截断字符串的右边以与目标列的最大宽度相匹配。如果 ASC *截断空白*选项有效，那么将去掉原始字符串或被截断字符串的尾部空格。
VARGRAPHIC（仅适用于 DBCS）	偶数字节的字符串。奇数字节的字符串无效并且是不可接受的。必要时，将截断有效字符串的右边部分以与目标列的最大宽度相匹配。

示例 ASC 文件

以下是一个 ASC 文件示例。每一行都以换行符序列结尾（在 Windows 操作系统上，每一行都以回车符/换行符序列结尾）。

```
Smith, Bob       4973     15.46
Jones, Suzanne   12345    16.34
Williams, Sam    452123  193.78
```

注:

1. 假定 ASC 文件不包含列名。
2. 字符串未用定界符括起来。ASC 文件中的列数据类型由数据库表中的目标列数据类型确定。
3. 在以下情况下，NULL 将导入到可空数据库列中:
   - 数字、DATE、TIME 或 TIMESTAMP 数据库列的目标是空白字段
   - 指定了没有开始和结束位置对的字段
   - 指定了开始和结束位置等于零的位置对
   - 数据行太短，无法包含对目标列有效的值
   - 使用了 NULL INDICATORS 装入选项，并且在空指示符列中发现 N（或用户指定的其他值）。
4. 如果目标列不可空，那么尝试将空白字段导入到数字、DATE、TIME 或 TIMESTAMP 列将导致拒绝该行。
5. 如果输入数据与目标列不兼容并且该列可空，那么会导入空值或拒绝该行，这取决于检测到错误的位置。如果该列不可空，那么拒绝该行。将会向消息文件写入消息，指定发现了不兼容的情况。

IXF 文件格式的 PC 版本

PC 版本的 IXF（PC/IXF）文件格式是数据库管理器对集成交换格式（IXF）数据交换体系结构的改编。IXF 体系结构专门用来允许关系数据库结构与数据的交换。PC/IXF 体系结构允许数据库管理器导出数据库，而不必理会接收产品的需求和特定反应。同样，导入 PC/IXF 文件的产品只需要理解 PC/IXF 体系结构；导出该文件的产品的特征是无关紧要的。PC/IXF 文件体系结构保持了导出和导入数据库系统的独立性。

IXF 体系结构是一种通用关系数据库交换格式，它支持很多关系数据类型，包括特定关系数据库产品可能不支持的某些类型。PC/IXF 文件格式保留了这一灵活性；例如，PC/IXF 体系结构同时支持单字节字符串（SBCS）和双字节字符串（DBCS）数据类型。并非所有实现都支持所有 PC/IXF 数据类型；但是，即使受限制的实现也会在导入时检测并处置不受支持的数据类型。

通常 PC/IXF 文件包含连续的变长记录序列。该文件按显示的顺序包含下列记录类型:

- 一个记录类型为 H 的头记录
- 一个记录类型为 T 的表记录
- 多个记录类型为 C 的列描述符记录（表中每列一个记录）
- 多个记录类型为 D 的数据记录（表中每行由一个或多个 D 记录表示）。

在出现 H 记录之后的任何位置，PC/IXF 文件还可能包含记录类型为 A 的应用程序记录。PC/IXF 文件中允许存在这些记录，以使应用程序能够在 PC/IXF 文件中包括未按 PC/IXF 格式定义的其他数据。对于读取 PC/IXF 文件的任何程序，如果不了解应用程序标识在 A 记录中代表的数据格式和内容，那么会忽略 A 记录。

PC/IXF 文件中的每个记录以记录长度指示符开头。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。读取 PC/IXF 文件的程序应使用这些记录长度来找到当前记录的结尾和下一条记录的开头。H、T 和 C 记录必须足够大以包括它们

定义的所有字段，当然，它们的记录长度字段必须与实际长度一致。但是，如果要在其中一个记录的结尾添加其他数据（如*新*字段），那么读取 PC/IXF 文件的预先存在程序应忽略其他数据，最多生成一条警告消息。但是，对于写入 PC/IXF 文件的程序，应写入其精确长度需要包含所有定义的字段的 H、T 和 C 记录。

如果 PC/IXF 文件包含 LOB 位置说明符（LLS）列，那么每个 LLS 列必须有自己的 D 记录。D 记录将由 export 实用程序自动创建，但如果使用第三方工具来生成 PC/IXF 文件，那么需要手动将它们创建。而且，LLS 是表中的每个 LOB 列所必需的，包括带有空值的列。如果 LOB 列为空，那么需要创建表示空 LOB 的 LLS。

每个 XML 列的 D 记录条目将包含 2 个字节的小尾数法来指示 XML 数据说明符（XDS）的长度，后面紧跟该 XDS 本身。

例如，以下 XDS:

```
XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

在 D 记录中将由以下字节表示:

```
0x3D 0x00 XDS FIL="a.xml" OFF="1000" LEN="100" SCH="RENATA.SCHEMA" />
```

PC/IXF 文件记录由包含字符数据的字段组成。import 实用程序和 export 实用程序使用目标数据库的 CPGID 解释此字符数据，但有两处例外:

- A 记录的 IXFADATA 字段。

  IXFADATA 字段中包含的字符数据的代码页环境是由创建和处理特定 A 记录的应用程序确定的；即环境随实施的不同而变化。
- D 记录的 IXFDCOLS 字段。

  IXFDCOLS 字段中包含的字符数据的代码页环境是由 C 记录中包含的信息决定的，C 记录定义特定列及其数据。

H、T 和 C 记录中的数字字段以及 D 记录和 A 记录中的前缀部分应该是向右对齐的单字节字符，用于表示整数值，并且用前导零和空格填充。值零应该至少用一个（向右对齐的）零字符指示，不能全部用空格指示。只要未在数据类型暗示长度的位置使用其中一个数字字段（如 IXFCLENG），就应该对其填充空格。这些数字字段包括:

```
IXFHRECL, IXFTRECL, IXFCRECL, IXFDRECL, IXFARECL,
IXFHHCNT, IXFHSBCP, IXFHDBCP, IXFTCCNT, IXFTNAML,
IXFCLENG, IXFCDRID, IXFCPOSN, IXFCNAML, IXFCTYPE,
IXFCSBCP, IXFCDBCP, IXFCNDIM, IXFCDSIZ, IXFDRID
```

注: 数据库管理器 PC/IXF 文件格式与 System/370™ 不同。

PC/IXF 记录类型

有五种基本 PC/IXF 记录类型:

- 头
- 表
- 列描述符
- 数据
- 应用程序

和六个应用程序子类型供 DB2 使用:

- 索引
- 层次结构
- 子表
- 延续
- 终止
- 标识

每个 PC/IXF 记录类型被定义为一个字段序列，这些字段是必需的，并且必须按显示的顺序出现。

HEADER RECORD

FIELD NAME	LENGTH	TYPE	COMMENTS
IXFHRECL	06-BYTE	CHARACTER	记录长度
IXFHRECT	01-BYTE	CHARACTER	记录类型 ="H"
IXFHID	03-BYTE	CHARACTER	IXF 标识
IXFHVERS	04-BYTE	CHARACTER	IXF 版本
IXFHPROD	12-BYTE	CHARACTER	产品
IXFHDATE	08-BYTE	CHARACTER	写入的日期
IXFHTIME	06-BYTE	CHARACTER	写入的时间
IXFHHCNT	05-BYTE	CHARACTER	首部记录计数
IXFHSBCP	05-BYTE	CHARACTER	单字节代码页
IXFHDBCP	05-BYTE	CHARACTER	双字节代码页
IXFHFIL1	02-BYTE	CHARACTER	保留

下列字段包含在头记录中:

IXFHRECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。H 记录必须足够长以包括定义的所有字段。

IXFHRECT

IXF 记录类型，对此记录设置为 H。

IXFHID

文件格式标识，对此文件设置为 IXF。

IXFHVERS

创建文件时使用的 PC/IXF 格式级别，设置为“0002”。

IXFHPROD

可由创建文件的程序用于标识自身的字段。如果已填充此字段，那么前六个字节将用于标识创建该文件的产品，后六个字节用于指示创建产品的版本或发行版。数据库管理器使用此字段来指示特定于数据库管理器的数据的存在。

IXFHDATE

写入文件的日期，格式为 *yyyymmdd*。

IXFHTIME

写入文件的时间，格式为 *hhmmss*。此字段是可选的，并且可以留为空白。

IXFHHCNT

此文件中第一个数据记录之前的 H、T 和 C 记录数。A 记录未包括在此计数中。

IXFHSBCP

单字节代码页字段，包含一个单字节字符，用于表示 SBCS CPGID 或“00000”。

export 实用程序将此字段设置为等于已导出数据库表的 SBCS CPGID。例如，如果表 SBCS CPGID 为 850，那么此字段包含“00850”。

IXFHDBCP

双字节代码页字段，包含一个单字节字符，用于表示 DBCS CPGID 或“00000”。

export 实用程序将此字段设置为等于已导出数据库表的 DBCS CPGID。例如，如果表 DBCS CPGID 为 301，那么此字段包含“00301”。

IXFHFIL1

空闲字段设置为两个空格以与主机 IXF 文件中的保留字段相匹配。

```
TABLE RECORD

      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     -------    ---------   -------------

      IXFTRECL       006-BYTE   CHARACTER   记录长度
      IXFTRECT       001-BYTE   CHARACTER   记录类型 ="T"
      IXFTNAML       003-BYTE   CHARACTER   名称长度
      IXFTNAME       256-BYTE   CHARACTER   数据名称
      IXFTQULL       003-BYTE   CHARACTER   限定符长度
      IXFTQUAL       256-BYTE   CHARACTER   限定符
      IXFTSRC        012-BYTE   CHARACTER   数据源
      IXFTDATA       001-BYTE   CHARACTER   数据约定 ="C"
      IXFTFORM       001-BYTE   CHARACTER   数据格式 ="M"
      IXFTMFRM       005-BYTE   CHARACTER   机器格式 ="PC"
      IXFTLOC        001-BYTE   CHARACTER   数据位置 ="I"
      IXFTCCNT       005-BYTE   CHARACTER   "C"记录计数
      IXFTFIL1       002-BYTE   CHARACTER   保留
      IXFTDESC       030-BYTE   CHARACTER   数据描述
      IXFTPKNM       257-BYTE   CHARACTER   主键名称
      IXFTDSPC       257-BYTE   CHARACTER   保留
      IXFTISPC       257-BYTE   CHARACTER   保留
      IXFTLSPC       257-BYTE   CHARACTER   保留
```

下列字段包含在表记录中:

IXFTRECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。T 记录必须足够长以包括定义的所有字段。

IXFTRECT

IXF 记录类型，对此记录设置为 T。

IXFTNAML

IXFTNAME 字段中的表名长度（以字节计）。

IXFTNAME

表的名称。如果每个文件只有一个表，那么这只是参考字段。数据库管理器在导入数据时不使用此字段。在写入 PC/IXF 文件时，数据库管理器会将 DOS 文件名（及可能的路径信息）写至此字段。

IXFTQULL

IXFTQUAL 字段中的表名限定符长度（以字节计）。

IXFTQUAL

表名限定符，标识关系系统中的表的创建者。这只是参考字段。如果写入文件

的程序没有要写至此字段的数据，那么首选填充值为空白。读取文件的程序可能打印或显示此字段，或者将它存储在参考字段中，但计算结果不应取决于此字段的内容。

IXFTSRC

用于指示数据的原始源。这只是参考字段。如果写入文件的程序没有要写至此字段的数据，那么首选填充值为空白。读取文件的程序可能打印或显示此字段，或者将它存储在参考字段中，但计算结果不应取决于此字段的内容。

IXFTDATA

用于描述数据的约定。对于导入和导出，此字段必须设置为 C，以指示各个列属性将在接来下的列描述符（C）记录中描述，并且数据遵循 PC/IXF 约定。

IXFTFORM

用于存储数字数据的约定。此字段必须设置为 M，指示数据（D）记录中的数字数据按 IXFTMFRM 字段指定的机器（内部）格式存储。

IXFTMFRM

PC/IXF 文件中的任何机器数据的格式。如果此字段设置为 PC*bbb*，那么 数据库管理器仅读取或写入文件，其中 *b* 表示空白，而 PC 指定 PC/IXF 文件中的数据使用 IBM PC 机器格式。

IXFTLOC

数据的位置。数据库管理器仅支持值 I，表示该数据在此文件内部。

IXFTCCNT

此表中的 C 记录数。这是一个向右对齐的字符，用于表示整数值。

IXFTFIL1

空闲字段设置为两个空格以与主机 IXF 文件中的保留字段相匹配。

IXFTDESC

有关该表的描述性数据。这只是参考字段。如果写入文件的程序没有要写至此字段的数据，那么首选填充值为空白。读取文件的程序可能打印或显示此字段，或者将它存储在参考字段中，但计算结果不应取决于此字段的内容。如果该列不为空并插入了缺省值，并且表名来自工作站数据库，那么此字段包含 NOT NULL WITH DEFAULT。

IXFTPKNM

对表定义的主键的名称（如果存在）。该名称存储为以 NULL 结束的字符串。

IXFTDSPC

保留此字段以供将来使用。

IXFTISPC

保留此字段以供将来使用。

IXFTLSPC

保留此字段以供将来使用。

COLUMN DESCRIPTOR RECORD

FIELD NAME	LENGTH	TYPE	COMMENTS
IXFCRECL	006-BYTE	CHARACTER	记录长度
IXFCRECT	001-BYTE	CHARACTER	记录类型 ="C"
IXFCNAML	003-BYTE	CHARACTER	列名长度
IXFCNAME	256-BYTE	CHARACTER	列名
IXFCNULL	001-BYTE	CHARACTER	列允许空

```
IXFCDEF        001-BYTE   CHARACTER   列具有缺省值
IXFCSLCT       001-BYTE   CHARACTER   列选中标志
IXFCKPOS       002-BYTE   CHARACTER   主键中的位置
IXFCCLAS       001-BYTE   CHARACTER   数据类
IXFCTYPE       003-BYTE   CHARACTER   数据类型
IXFCSBCP       005-BYTE   CHARACTER   单字节代码页
IXFCDBCP       005-BYTE   CHARACTER   双字节代码页
IXFCLENG       005-BYTE   CHARACTER   列数据长度
IXFCDRID       003-BYTE   CHARACTER   "D"记录标识
IXFCPOSN       006-BYTE   CHARACTER   列位置
IXFCDESC       030-BYTE   CHARACTER   列描述
IXFCLOBL       020-BYTE   CHARACTER   LOB 列长度
IXFCUDTL       003-BYTE   CHARACTER   UDT 名称长度
IXFCUDTN       256-BYTE   CHARACTER   UDT 名称
IXFCDEFL       003-BYTE   CHARACTER   缺省值长度
IXFCDEFV       254-BYTE   CHARACTER   缺省值
IXFCREF        001-BYTE   CHARACTER   引用类型
IXFCNDIM       002-BYTE   CHARACTER   维数
IXFCDSIZ       varying    CHARACTER   每个维的大小
```

下列字段包含在列描述符记录中:

IXFCRECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。C 记录必须足够长以包括定义的所有字段。

IXFCRECT

IXF 记录类型，对此记录设置为 C。

IXFCNAML

IXFCNAME 字段中的列名长度（以字节计）。

IXFCNAME

列的名称。

IXFCNULL

指定此列中是否允许使用空值。有效设置为 Y 或 N。

IXFCDEF

指定是否对此字段定义了缺省值。有效设置为 Y 或 N。

IXFCSLCT

一个过时字段，本来打算用于允许选择数据中的某个列子集。写入 PC/IXF 文件的程序应总是在此字段中存储 Y。读取 PC/IXF 文件的程序应忽略此字段。

IXFCKPOS

主键中包含的列的位置。有效值范围在 01 到 16 之间，或者如果列未包括在主键中，那么有效值为 N。

IXFCCLAS

要在 IXFCTYPE 字段中使用的数据类型的类。数据库管理器仅支持关系类型（R）。

IXFCTYPE

列的数据类型。

IXFCSBCP

包含一个单字节字符，用于表示 SBCS CPGID。此字段对单字节字符数据指定 CPGID，这种情况经常出现在此列的 D 记录的 IXFDCOLS 字段中。

此字段的语义随列数据类型（在 IXFCTYPE 字段中指定）不同而变化。

- 对于字符串列，此字段通常应包含等于 H 记录中的 IXFHSBCP 字段的非零值，但也允许包含其他值。如果此值为零，那么该列将解释为包含二进制位串数据。
- 对于数字列，此字段没有意义。它将被 export 实用程序设置为零，并且被 import 实用程序忽略。
- 对于日期或时间列，此字段没有意义。它将被 export 实用程序设置为 IXFHSBCP 字段的值，并且被 import 实用程序忽略。
- 对于图形列，此字段必须为零。

IXFCDBCP

包含一个单字节字符，用于表示 DBCS CPGID。此字段对双字节字符数据指定 CPGID，这种情况经常出现在此列的 D 记录的 IXFDCOLS 字段中。

此字段的语义随列数据类型（在 IXFCTYPE 字段中指定）不同而变化。

- 对于字符串列，此字段通常应该为零，或者包含等于 H 记录中的 IXFHDBCP 字段的值；但也允许包含其他值。如果 IXFCSBCP 字段中的值为零，那么此字段中的值必须为零。
- 对于数字列，此字段没有意义。它将被 export 实用程序设置为零，并且被 import 实用程序忽略。
- 对于日期或时间列，此字段没有意义。它将被 export 实用程序设置为零，并且被 import 实用程序忽略。
- 对于图形列，此字段的值必须等于 IXFHDBCP 字段的值。

IXFCLENG

提供有关要描述的列的大小信息。对于某些数据类型，此字段未被使用并且应该包含空白。对于其他数据类型，此字段包含向右对齐的字符，用于表示指定列长度的整数。对于其他数据类型，此字段分为两个子字段：3 字节用于精度，2 字节用于小数位；这些子字段都是向右对齐的字符，用于表示整数。

IXFCDRID

D 记录标识。此字段包含向右对齐的字符，用于表示整数值。可在 PC/IXF 文件中使用若干 D 记录来包含每行数据。此字段指定用于数据行的若干 D 记录中有哪些 D 记录包含列数据。值 1（如 001）指示某列的数据在行数据的第一个 D 记录中。第一个 C 记录的 IXFCDRID 值必须为 1。所有后续 C 记录的 IXFCDRID 值必须等于之前 C 记录中的值或更高的值。

IXFCPOSN

此字段中的值用于找到列数据，该列数据在表示表数据行的其中一个 D 记录中。这是此列数据在 D 记录的 IXFDCOLS 字段中的起始位置。如果该列可空，那么 IXFCPOSN 指向空指示符；否则它将指向数据本身。如果某列包含变长数据，那么数据本身以当前长度指示符开头。D 记录的 IXFDCOLS 字段中的第一个字节的 IXFCPOSN 值为 1（而不是 0）。如果某列在新的 D 记录中，那么 IXFCPOSN 的值应该为 1；否则 IXFCPOSN 值应该逐列递增以使数据值不会重叠。

IXFCDESC

有关该列的描述性信息。这只是参考字段。如果写至文件的程序没有要写至此字段的数据，那么首选填充值为空白。读取文件的程序可能打印或显示此字段，或者将它存储在参考字段中，但计算结果不应取决于此字段的内容。

IXFCLOBL

此列中定义的长整型数据或 LOB 的长度（以字节计）。如果此列不是长整型数据或 LOB，那么此字段中的值为 000。

IXFCUDTL

IXFCUDTN 字段中的用户定义的类型（UDT）名称长度（以字节计）。如果此列类型不是 UDT，那么此字段中的值为 000。

IXFCUDTN

用户定义的类型的名称，该类型用作此列的数据类型。

IXFCDEFL

IXFCDEFV 字段中的缺省值长度（以字节计）。如果此列没有缺省值，那么此字段中的值为 000。

IXFCDEFV

如果已经定义了此列，那么对其指定缺省值。

IXFCREF

如果该列包含在层次结构中，那么此字段指定该列是数据列（D）还是引用列（R）。

IXFCNDIM

列中的维数。此版本的 PC/IXF 不支持数组。因此，此字段必须包含表示零整数值的字符。

IXFCDSIZ

每个维的大小或范围。此字段的长度为每个维 5 个字节。因为不支持数组（即维数必须为零），所以此字段的长度为零，并且实际上不存在。

```
DATA RECORD

      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     -------   ----------   -------------
      IXFDRECL       06-BYTE   CHARACTER   记录长度
      IXFDRECT       01-BYTE   CHARACTER   记录类型 ="D"
      IXFDRID        03-BYTE   CHARACTER   "D"记录标识
      IXFDFIL1       04-BYTE   CHARACTER   保留
      IXFDCOLS       varying   variable    列数据
```

下列字段包含在数据记录中:

IXFDRECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 D 记录必须足够长，以包括存储在记录中的当前出现的最后一个数据列的所有重要数据。

IXFDRECT

IXF 记录类型，对此记录设置为 D，指示它包含表的数据值。

IXFDRID

记录标识，标识用于数据行的若干 D 记录的序列内的特定 D 记录。对于数据行中的第一个 D 记录，此字段的值为 1；对于数据行中的第二个 D 记录，此字段的值为 2，以此类推。在每个数据行中，C 记录中调出的所有 D 记录标识都必须实际存在。

IXFDFIL1

空闲字段设置为四个空格以与保留字段相匹配；并且在主机 IXF 文件中占用一个位置以供可能出现的 shift-out 字符使用。

IXFDCOLS

列数据区域。数据记录（D 记录）的数据区由一个或多个列条目组成。每个列描述符记录对应一个列条目，它的 D 记录标识与 D 记录相同。在 D 记录中，列条目的起始位置由 C 记录中的 IXFCPOSN 值指示。

列条目数据的格式取决于该列是否可空:

- 如果该列可空（IXFCNULL 字段设置为 Y），那么列条目数据包括空指示符。如果该列不为空，那么指示符后跟特定于数据类型的信息，包括实际数据库值。空指示符是由两个字节组成的值，如果非空，那么设置为 x'0000'，如果为空，那么设置为 x'FFFF'。
- 如果该列不可空，那么列条目数据仅包括特定于数据类型的信息，包括实际数据库值。

对于变长数据类型，特定于数据类型的信息包括当前长度指示符。当前长度指示符是由两个字节组成的整数，格式由 IXFTMFRM 字段指定。

D 记录的数据区的长度不能超过 32 771 个字节。

```
APPLICATION RECORD

      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     -------   ---------   -------------
      IXFARECL       06-BYTE   CHARACTER   记录长度
      IXFARECT       01-BYTE   CHARACTER   记录类型 ="A"
      IXFAPPID       12-BYTE   CHARACTER   应用程序标识
   IXFADATA       varying   variable    特定于应用程序的数据
```

下列字段包含在应用程序记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，标识创建 A 记录的应用程序。数据库管理器创建的 PC/IXF 文件可以具有 A 记录，并且此字段的前 6 个字符设置为常量以标识数据库管理器，后 6 个字符标识数据库管理器的发行版或版本或者写入 A 记录的另一应用程序。

IXFADATA

此字段包含依赖于应用程序的补充数据，其格式和内容只有创建 A 记录的程序和可能处理 A 记录的其他应用程序才知道。

```
DB2 INDEX RECORD
      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
      IXFARECL       006-BYTE   CHARACTER   记录长度
      IXFARECT       001-BYTE   CHARACTER   记录类型 ="A"
```

```
   IXFAPPID       012-BYTE   CHARACTER   应用程序标识 =
                                         "DB2    02.00"
   IXFAITYP       001-BYTE   CHARACTER   特定于应用程序的数据类型 ="I"
IXFADATE       008-BYTE   CHARACTER   从"H"记录写入的日期
   IXFATIME       006-BYTE   CHARACTER   从"H"记录写入的时间
   IXFANDXL       002-BYTE   SHORT INT   索引名称长度
   IXFANDXN       256-BYTE   CHARACTER   索引名称
   IXFANCL        002-BYTE   SHORT INT   索引创建者名称长度
   IXFANCN        256-BYTE   CHARACTER   索引创建者名称
   IXFATABL       002-BYTE   SHORT INT   表名的长度
   IXFATABN       256-BYTE   CHARACTER   表名
   IXFATCL        002-BYTE   SHORT INT   表创建者名称长度
   IXFATCN        256-BYTE   CHARACTER   表创建者名称
   IXFAUNIQ       001-BYTE   CHARACTER   唯一规则
   IXFACCNT       002-BYTE   CHARACTER   列计数
   IXFAREVS       001-BYTE   CHARACTER   允许逆向扫描标志
   IXFAPCTF       002-BYTE   CHARACTER   可用 PCT 量
   IXFAPCTU       002-BYTE   CHARACTER   至少要使用的 PCT 量
   IXFAEXTI       001-BYTE   CHARACTER   保留
   IXFACNML       002-BYTE   SHORT INT   列名称长度
   IXFACOLN       不定       CHARACTER   索引中的列名称
```

将对每个用户定义的索引指定此类型的一个记录。此记录放在该表的所有 C 记录之后。下列字段包含在 DB2 索引记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，它将 DB2 标识为创建此 A 记录的应用程序。

IXFAITYP

指定这是 DB2 应用程序记录的子类型“I”。

IXFADATE

写入文件的日期，格式为 *yyyymmdd*。此字段的值必须与 IXFHDATE 的值相同。

IXFATIME

写入文件的时间，格式为 *hhmmss*。此字段的值必须与 IXFHTIME 的值相同。

IXFANDXL

IXFANDXN 字段中的索引名称长度（以字节计）。

IXFANDXN

索引的名称。

IXFANCL

IXFANCN 字段中的索引创建者名称长度（以字节计）。

IXFANCN

索引创建者名称。

IXFATABL

IXFATABN 字段中的表名长度（以字节计）。

IXFATABN

表的名称。

IXFATCL

IXFATCN 字段中的表创建者名称长度（以字节计）。

IXFATCN

表创建者的名称。

IXFAUNIQ

指定索引类型。对于主键，有效值为 P；对于唯一索引，有效值为 U；对于非唯一索引，有效值为 D。

IXFACCNT

指定索引定义中的列数。

IXFAREVS

指定是否允许对此索引执行逆向扫描。如果允许逆向扫描，那么有效值为 Y；如果不允许逆向扫描，那么有效值为 N。

IXFAPCTF

指定释放时要保留的索引页百分比。有效值范围是 -1 到 99。如果指定了值 -1，那么使用系统缺省值。

IXFAPCTU

指定合并两个索引页之前必须释放的最小索引页百分比。有效值范围是 00 到 99。

IXFAEXTI

保留以备将来使用。

IXFACNML

IXFACOLN 字段中的列名长度（以字节计）。

IXFACOLN

此索引包含的列的名称。有效值的格式为 *+name-name...*，其中 + 指定对列进行升序排序，而 - 指定对列进行降序排序。

```
DB2 HIERARCHY RECORD

      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
      IXFARECL       006-BYTE   CHARACTER   记录长度
      IXFARECT       001-BYTE   CHARACTER   记录类型 ="A"
      IXFAPPID       012-BYTE   CHARACTER   应用程序标识 =
                                           "DB2    02.00"
      IXFAXTYP       001-BYTE   CHARACTER   特定于应用程序的数据类型 ="X"
   IXFADATE       008-BYTE   CHARACTER   从"H"记录写入的日期
      IXFATIME       006-BYTE   CHARACTER   从"H"记录写入的时间
      IXFAYCNT       010-BYTE   CHARACTER   此层次结构的"Y"记录计数
      IXFAYSTR       010-BYTE   CHARACTER   此层次结构的起始列
```

将使用此类型的一个记录来描述层次结构。所有子表记录（请参阅下面的内容）都必须紧跟在层次结构记录之后，并且层次结构记录在表的所有 C 记录之后。下列字段包含在 DB2 层次结构记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，它将 DB2 标识为创建此 A 记录的应用程序。

IXFAXTYP

指定这是 DB2 应用程序记录的子类型“X”。

IXFADATE

写入文件的日期，格式为 *yyyymmdd*。此字段的值必须与 IXFHDATE 的值相同。

IXFATIME

写入文件的时间，格式为 *hhmmss*。此字段的值必须与 IXFHTIME 的值相同。

IXFAYCNT

指定应该放在此层次结构记录之后的子表记录数。

IXFAYSTR

指定已导出数据的起始位置处的子表记录的索引。如果层次结构的导出从非根子表开始，那么会导出此子表的所有父表。此子表在 IXF 文件内的位置也存储在此字段中。第一个 X 记录表示索引为零的列。

```
DB2 SUBTABLE RECORD

      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
      IXFARECL       006-BYTE   CHARACTER   记录长度
      IXFARECT       001-BYTE   CHARACTER   记录类型 ="A"
      IXFAPPID       012-BYTE   CHARACTER   应用程序标识 =
                                           "DB2    02.00"
      IXFAYTYP       001-BYTE   CHARACTER   特定于应用程序的数据类型 =
                                           "Y"
      IXFADATE       008-BYTE   CHARACTER   从"H"记录写入的日期
      IXFATIME       006-BYTE   CHARACTER   从"H"记录写入的时间
      IXFASCHL       003-BYTE   CHARACTER   类型模式名长度
      IXFASCHN       256-BYTE   CHARACTER   类型模式名
      IXFATYPL       003-BYTE   CHARACTER   类型名称长度
      IXFATYPN       256-BYTE   CHARACTER   类型名
      IXFATABL       003-BYTE   CHARACTER   表名长度
      IXFATABN       256-BYTE   CHARACTER   表名
      IXFAPNDX       010-BYTE   CHARACTER   父表的子表索引
      IXFASNDX       005-BYTE   CHARACTER   当前表的起始列索引
IXFAENDX       005-BYTE   CHARACTER   当前表的结束列索引
```

将使用此类型的一个记录来描述层次结构中的某个子表。属于层次结构的所有子表记录必须存储在一起，并且紧跟在相应层次结构记录之后。子表由一个或多个列组成，每个列将在一个列记录中描述。子表中的每个列必须在一组连续的 C 记录中作出描述。下列字段包含在 DB2 子表记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整

数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，它将 DB2 标识为创建此 A 记录的应用程序。

IXFAYTYP

指定这是 DB2 应用程序记录的子类型“Y”。

IXFADATE

写入文件的日期，格式为 *yyyymmdd*。此字段的值必须与 IXFHDATE 的值相同。

IXFATIME

写入文件的时间，格式为 *hhmmss*。此字段的值必须与 IXFHTIME 的值相同。

IXFASCHL

IXFASCHN 字段中的子表模式名长度（以字节计）。

IXFASCHN

子表模式的名称。

IXFATYPL

IXFATYPN 字段中的子表名长度（以字节计）。

IXFATYPN

子表的名称。

IXFATABL

IXFATABN 字段中的表名长度（以字节计）。

IXFATABN

表的名称。

IXFAPNDX

父代子表的子表记录索引。如果此子表是层次结构的根，那么此字段包含值 -1。

IXFASNDX

构成此子表的列记录的起始索引。

IXFAENDX

构成此子表的列记录的结尾索引。

DB2 CONTINUATION RECORD

FIELD NAME	LENGTH	TYPE	COMMENTS
IXFARECL	006-BYTE	CHARACTER	记录长度
IXFARECT	001-BYTE	CHARACTER	记录类型 ="A"
IXFAPPID	012-BYTE	CHARACTER	应用程序标识 = "DB2 02.00"
IXFACTYP	001-BYTE	CHARACTER	特定于应用程序的数据类型 ="C"
IXFADATE	008-BYTE	CHARACTER	从"H"记录写入的日期
IXFATIME	006-BYTE	CHARACTER	从"H"记录写入的时间

```
   IXFALAST       002-BYTE   SHORT INT   上一个软盘卷编号
   IXFATHIS       002-BYTE   SHORT INT   此软盘卷编号
   IXFANEXT       002-BYTE   SHORT INT   下一个软盘卷编号
```

此记录可在多卷 IXF 文件中的每个文件的结尾找到，除非该文件是最后一卷；还可在多卷 IXF 文件中的每个文件的开头找到它，除非该文件是第一卷。此记录用于记录文件顺序。下列字段包含在 DB2 持续记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，它将 DB2 标识为创建此 A 记录的应用程序。

IXFACTYP

指定这是 DB2 应用程序记录的子类型“C”。

IXFADATE

写入文件的日期，格式为 *yyyymmdd*。此字段的值必须与 IXFHDATE 的值相同。

IXFATIME

写入文件的时间，格式为 *hhmmss*。此字段的值必须与 IXFHTIME 的值相同。

IXFALAST

此字段是二进制字段，采用小尾数法格式。该值应该比 IXFATHIS 中的值小。

IXFATHIS

此字段是二进制字段，采用小尾数法格式。此字段中针对连续卷的值也应该是连续的。第一卷的值为 1。

IXFANEXT

此字段是二进制字段，采用小尾数法格式。该值应该比 IXFATHIS 中的值小，除非该记录在文件开头，这种情况下该值应该为零。

```
DB2 TERMINATE RECORD

      FIELD NAME     LENGTH     TYPE        COMMENTS
   ----------     --------   ---------   -------------
      IXFARECL       006-BYTE   CHARACTER   记录长度
      IXFARECT       001-BYTE   CHARACTER   记录类型 ="A"
      IXFAPPID       012-BYTE   CHARACTER   应用程序标识 =
                                            "DB2    02.00"
      IXFAETYP       001-BYTE   CHARACTER   特定于应用程序的数据类型 ="E"
   IXFADATE       008-BYTE   CHARACTER   从"H"记录写入的日期
      IXFATIME       006-BYTE   CHARACTER   从"H"记录写入的时间
```

此记录是位于 IXF 文件结尾处的文件末尾标记。下列字段包含在 DB2 终止记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整

数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，它将 DB2 标识为创建此 A 记录的应用程序。

IXFAETYP

指定这是 DB2 应用程序记录的子类型“E”。

IXFADATE

写入文件的日期，格式为 *yyyymmdd*。此字段的值必须与 IXFHDATE 的值相同。

IXFATIME

写入文件的时间，格式为 *hhmmss*。此字段的值必须与 IXFHTIME 的值相同。

DB2 IDENTITY RECORD

FIELD NAME	LENGTH	TYPE	COMMENTS
IXFARECL	06-BYTE	CHARACTER	记录长度
IXFARECT	01-BYTE	CHARACTER	记录类型 ="A"
IXFAPPID	12-BYTE	CHARACTER	应用程序标识
IXFATYPE	01-BYTE	CHARACTER	特定于应用程序的记录类型 ="S"
IXFADATE	08-BYTE	CHARACTER	应用程序记录创建日期
IXFATIME	06-BYTE	CHARACTER	应用程序记录创建时间
IXFACOLN	06-BYTE	CHARACTER	标识列的列号
IXFAITYP	01-BYTE	CHARACTER	总是生成（"Y"或"N"）
IXFASTRT	33-BYTE	CHARACTER	标识 START AT 值
IXFAINCR	33-BYTE	CHARACTER	标识 INCREMENT BY 值
IXFACACH	10-BYTE	CHARACTER	标识 CACHE 值
IXFAMINV	33-BYTE	CHARACTER	标识 MINVALUE
IXFAMAXV	33-BYTE	CHARACTER	标识 MAXVALUE
IXFACYCL	01-BYTE	CHARACTER	标识 CYCLE（"Y"或"N"）
IXFAORDR	01-BYTE	CHARACTER	标识 ORDER（"Y"或"N"）
IXFARMRL	03-BYTE	CHARACTER	标识注释长度
IXFARMRK	254-BYTE	CHARACTER	标识注释值

下列字段包含在 DB2 标识记录中:

IXFARECL

记录长度指示符。它由 6 个字节组成，以向右对齐的字符来表示整数值，该整数值指定记录长度指示符之后的 PC/IXF 记录部分的长度（以字节计），即总记录大小减去 6 个字节。每个 A 记录必须足够长以至少包括整个 IXFAPPID 字段。

IXFARECT

IXF 记录类型，对此记录设置为 A，指示这是一个应用程序记录。如果程序不太清楚应用程序标识代表的数据的内容和格式，那么将忽略这些记录。

IXFAPPID

应用程序标识，它将 DB2 标识为创建此 A 记录的应用程序。

IXFATYPE

特定于应用程序的记录类型。此字段的值应该一直是“S”。

IXFADATE

写入文件的日期，格式为 *yyyymmdd*。此字段的值必须与 IXFHDATE 的值相同。

IXFATIME

写入文件的时间，格式为 *hhmmss*。此字段的值必须与 IXFHTIME 的值相同。

IXFACOLN

表中的标识列的列号。

IXFAITYP

标识列的类型。值“Y”指示标识列一直是 GENERATED。所有其他值的解释意味着该列的类型为 GENERATED BY DEFAULT。

IXFASTRT

标识列的 START AT® 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。

IXFAINCR

标识列的 INCREMENT BY 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。

IXFACACH

标识列的 CACHE 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。值“1”对应 NO CACHE 选项。

IXFAMINV

标识列的 MINVALUE 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。

IXFAMAXV

标识列的 MAXVALUE 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。

IXFACYCL

标识列的 CYCLE 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。值“Y”对应于 CYCLE 选项，而任何其他值对应于 NO CYCLE。

IXFAORDR

标识列的 ORDER 值，该标识列是在创建表时提供给 CREATE TABLE 语句的。值“Y”对应于 ORDER 选项，而任何其他值对应于 NO ORDER。

IXFARMRL

IXFARMRK 字段中的注释长度（以字节计）。

IXFARMRK

这是用户输入的与标识列相关联的注释。这只是参考字段。数据库管理器在导入数据时不使用此字段。

PC/IXF 数据类型

表 52. PC/IXF 数据类型

名称	IXFCTYPE 值	描述
BIGINT	492	一个 8 字节整数，格式由 IXFTMFRM 指定。它表示 -9 223 372 036 854 775 808 与 9 223 372 036 854 775 807 之间的整数。IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。IXFCLENG 不会被使用，应该包含空白。
BLOB 和 CLOB	404 和 408	变长字符串。字符串的最大长度包含在列描述符记录的 IXFCLENG 字段中，并且不能超过 32 767 字节。字符串本身会加上当前长度指示符作为前缀，这是一个 4 字节整数，用于指定字符串的长度（以字节计）。该字符串在由 IXFCSBCP 指示的代码页中。 以下描述仅适用于 BLOB: 如果 IXFCSBCP 为零，那么字符串为位数据，并且不应被任何变换程序进行转换。 以下描述仅适用于 CLOB: 如果 IXFCDBCP 不为零，那么字符串还可包含 IXFCDBCP 指示的代码页中的双字节字符。
BLOB_LOCATION _ SPECIFIER 和 DBCLOB_ LOCA-TION_ SPECIFIER	960、964 和 968	定长字段，不能超过 255 个字节。LOB 位置说明符（LLS）在 IXFCSBCP 指示的代码页中。如果 IXFCSBCP 为零，那么 LLS 为位数据，并且不应被任何变换程序进行转换。如果 IXFCDBCP 不为零，那么字符串还可包含 IXFCDBCP 指示的代码页中的双字节字符。 因为 LLS 的长度存储在 IXFCLENG 中，所以原始 LOB 的实际长度将会丢失。因为将使用 LLS 的长度创建 LOB，所以不应使用带有此类型的列的 PC/IXF 文件重新创建 LOB 字段。
BLOB_FILE、CLOB_FILE 和 DBCLOB_FILE	916、920 和 924	包含 SQLFILE 结构并且填充了 *name_length* 和 *name* 字段的定长字段。该结构的长度包含在列描述符记录的 IXFCLENG 字段中，并且不能超过 255 个字节。该文件名在由 IXFCSBCP 指示的代码页中。如果 IXFCDBCP 不为零，那么文件名还可包含 IXFCDBCP 指示的代码页中的双字节字符。如果 IXFCSBCP 为零，那么文件名为位数据，并且不应被任何变换程序进行转换。 因为结构的长度存储在 IXFCLENG 中，所以原始 LOB 的实际长度将会丢失。因为将使用长度 *sql_lobfile_len* 创建 LOB，所以不应使用带有类型为 BLOB_FILE、CLOB_FILE 或 DBCLOB_FILE 的列的 IXF 文件重新创建 LOB 字段。

表 52. PC/IXF 数据类型 (续)

名称	IXFCTYPE 值	描述
CHAR	452	定长字符串。该字符串长度包含在列描述符记录的 IXFCLENG 字段中，并且不能超过 254 个字节。该字符串在由 IXFCSBCP 指示的代码页中。如果 IXFCDBCP 不为零，那么字符串还可包含 IXFCDBCP 指示的代码页中的双字节字符。如果 IXFCSBCP 为零，那么字符串为位数据，并且不应被任何变换程序进行转换。
DATE	384	符合格列高利历的时间点。每个日期是一个使用国际标准组织（ISO）格式的 10 字节字符串：*yyyy-mm-dd*。年份部分的范围为 0001 到 9999。月份部分的范围为 01 到 12。日期部分的范围为 01 到 *n*，其中 *n* 取决于月份，它们遵循每个月天数及闰年的通用规则。任何部分都不能省略前导零。IXFCLENG 不会被使用，应该包含空白。DATE 中的有效字符在所有 PC ASCII 代码页中都是不变的；因此 IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。
DBCLOB	412	双字节字符的变长字符串。列描述符记录中的 IXFCLENG 字段指定字符串中的最大双字节字符数，并且不能超过 16 383。字符串本身会加上当前长度指示符作为前缀，这是一个 4 字节整数，用于指定双字节字符串的长度（即，此整数的值是字符串长度的一半）。字符串在 C 记录的 IXFCDBCP 指定的 DBCS 代码页中。因为字符串仅包含双字节字符数据，所以 IXFCSBCP 应该为零。周围没有 shift-in 或 shift-out 字符。
DECIMAL	484	压缩十进制数，精度为 P（由列描述符记录中的 IXFCLENG 的前个三字节指定），标度为 S（由 IXFCLENG 的后两个字节指定）。压缩十进制数的长度（以字节计）为 (P+2)/2。精度必须是 1 到 31 之间的奇数（包括 1 和 31）。压缩十进制数使用 IXFTMFRM 指定的内部格式，其中 PC 的压缩十进制定义与 System/370 的压缩十进制相同。IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。
DECFLOAT	996	十进制浮点值是包含小数点的 IEEE 754r 数字。小数点的位置存储在每个十进制浮点值中。十进制浮点数的范围是精度为 16 位或 34 位的数字，其对应的指数范围分别为 10-383 至 10+384 或 10-6143 至 10+6144。16 位值的存储长度为 8 个字节，34 位值的存储长度为 16 个字节。

表 52. PC/IXF 数据类型 (续)

名称	IXFCTYPE 值	描述
FLOATING POINT	480	长（8 字节）或短（4 字节）浮点数，取决于 IXFCLENG 是设置为 8 还是 4。数据使用 IXFTMFRM 指定的内部机器格式。IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。数据库管理器不支持 4 字节浮点数。
GRAPHIC	468	双字节字符的定长字符串。列描述符记录中的 IXFCLENG 字段指定字符串中的双字节字符数，并且不能超过 127。字符串的实际长度是 IXFCLENG 字段值的两倍（以字节计）。字符串在 C 记录的 IXFCDBCP 指定的 DBCS 代码页中。因为字符串仅包含双字节字符数据，所以 IXFCSBCP 应该为零。周围没有 shift-in 或 shift-out 字符。
INTEGER	496	一个 4 字节整数，格式由 IXFTMFRM 指定。它表示范围在 -2 147 483 648 与 +2 147 483 647 之间的整数。IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。IXFCLENG 不会被使用，应该包含空白。
LONGVARCHAR	456	变长字符串。字符串的最大长度包含在列描述符记录的 IXFCLENG 字段中，并且不能超过 32 767 字节。字符串本身会加上当前长度指示符作为前缀，这是一个 2 字节整数，用于指定字符串的长度（以字节计）。该字符串在由 IXFCSBCP 指示的代码页中。如果 IXFCDBCP 不为零，那么字符串还可包含 IXFCDBCP 指示的代码页中的双字节字符。如果 IXFCSBCP 为零，那么字符串为位数据，并且不应被任何变换程序进行转换。
LONG VARGRAPHIC	472	双字节字符的变长字符串。列描述符记录中的 IXFCLENG 字段指定字符串的最大双字节字符数，并且不能超过 16 383。字符串本身会加上当前长度指示符作为前缀，这是一个 2 字节整数，用于指定双字节字符串的长度（即，此整数的值是字符串长度的一半）。字符串在 C 记录的 IXFCDBCP 指定的 DBCS 代码页中。因为字符串仅包含双字节字符数据，所以 IXFCSBCP 应该为零。周围没有 shift-in 或 shift-out 字符。
SMALLINT	500	一个 2 字节整数，格式由 IXFTMFRM 指定。它表示范围在 -32 768 到 +32 767 之间的整数。IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。IXFCLENG 不会被使用，应该包含空白。

表 52. PC/IXF 数据类型 (续)

名称	IXFCTYPE 值	描述
TIME	388	符合 24 小时时钟的时间点。每个时间都是使用 ISO 格式的 8 字节字符串: *hh.mm.ss*。小时部分的范围为 00 到 24，而其他部分的范围为 00 到 59。如果小时为 24，那么其他部分为 00。最小时间为 00.00.00，最大时间为 24.00.00。任何部分都不能省略前导零。IXFCLENG 不会被使用，应该包含空白。TIME 中的有效字符在所有 PC ASCII 代码页中都是不变的；因此 IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。
TIMESTAMP	392	微秒精度的日期和时间。每个时间戳记都是具有如下格式的字符串: *yyyy-mm-dd-hh.mm.ss.nnnnnn*（年月日小时分钟秒微秒）。IXFCLENG 不会被使用，应该包含空白。TIMESTAMP 中的有效字符在所有 PC ASCII 代码页中都是不变的；因此 IXFCSBCP 和 IXFCDBCP 没有意义，应该为零。
VARCHAR	448	变长字符串。字符串的最大长度包含在列描述符记录的 IXFCLENG 字段中，并且不能超过 254 个字节。字符串本身会加上当前长度指示符作为前缀，这是一个 2 字节整数，用于指定字符串的长度（以字节计）。该字符串在由 IXFCSBCP 指示的代码页中。如果 IXFCDBCP 不为零，那么字符串还可包含 IXFCDBCP 指示的代码页中的双字节字符。如果 IXFCSBCP 为零，那么字符串为位数据，并且不应被任何变换程序进行转换。
VARGRAPHIC	464	双字节字符的变长字符串。列描述符记录中的 IXFCLENG 字段指定字符串中的最大双字节字符数，并且不能超过 127。字符串本身会加上当前长度指示符作为前缀，这是一个 2 字节整数，用于指定双字节字符串的长度（即，此整数的值是字符串长度的一半）。字符串在 C 记录的 IXFCDBCP 指定的 DBCS 代码页中。因为字符串仅包含双字节字符数据，所以 IXFCSBCP 应该为零。周围没有 shift-in 或 shift-out 字符。

并非所有 PC/IXF 字符或图形列的 IXFCSBCP 和 IXFCDBCP 值的所有组合都有效。带有无效 (IXFCSBCP,IXFCDBCP) 组合的 PC/IXF 字符或图形列是无效数据类型。

表 53. 有效 PC/IXF 数据类型

PC/IXF 数据类型	有效 (IXFCSBCP,IXFCDBCP) 对	无效 (IXFCSBCP,IXFCDBCP) 对
CHAR、VARCHAR 或 LONG VARCHAR	(0,0)、(x,0) 或 (x,y)	(0,y)
BLOB	(0,0)	(x,0)、(0,y) 或 (x,y)

表 53. 有效 PC/IXF 数据类型 (续)

PC/IXF 数据类型	有效 (IXFCSBCP,IXFCDBCP) 对	无效 (IXFCSBCP,IXFCDBCP) 对
CLOB	(x,0) 和 (x,y)	(0,0) 和 (0,y)
GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC 或 DBCLOB	(0,y)	(0,0)、(x,0) 或 (x,y)
注: x 和 y 不为零。		

PC/IXF 数据类型描述

表 54. PC/IXF 文件格式可接受的数据类型格式

数据类型	export 实用程序创建的文件中的格式	import 实用程序可接受的格式
BIGINT	创建与数据库列完全相同的 BIGINT 列。	接受任何数字类型（SMALLINT、INTEGER、BIGINT、 DECIMAL 或 FLOAT）的列。如果各个值不在范围 -9 223 372 036 854 775 808 到 9 223 372 036 854 775 807 之间，那么会拒绝这些值。
BLOB	创建 PC/IXF BLOB 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	满足下列条件时，可接受 PC/IXF CHAR、VARCHAR、 LONG VARCHAR、BLOB、 BLOB_FILE 或 BLOB_LOCATION_SPECIFIER 列: • 数据库列被标记为“用于位数据” • PC/IXF 列单字节代码页值等于数据库列的 SBCS CPGID，并且 PC/IXF 列双字节代码页值等于零或数据库列的 DBCS CPGID。还可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC BLOB 列。如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。
CHAR	创建 PC/IXF CHAR 列。数据库列长度、SBCS CPGID 值和 DBCS CPGID 值将复制至 PC/IXF 列描述符记录。	满足下列条件时，可接受 PC/IXF CHAR、VARCHAR 或 LONG VARCHAR 列: • 数据库列被标记为“用于位数据” • PC/IXF 列单字节代码页值等于数据库列的 SBCS CPGID，并且 PC/IXF 列双字节代码页值等于零或数据库列的 DBCS CPGID。 如果数据库列标记为“用于位数据”，那么还可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC 列。在任何情况下，如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的长度相兼容。必要时将在数据的右边填充单字节空格（x'20'）。

表 54. PC/IXF 文件格式可接受的数据类型格式 (续)

数据类型	export 实用程序创建的文件中的格式	import 实用程序可接受的格式
CLOB	创建 PC/IXF CLOB 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	如果 PC/IXF 列单字节代码页值等于数据库列的 SBCS CPGID，并且 PC/IXF 列双字节代码页值等于零或数据库列的 DBCS CPGID，那么可以接受 PC/IXF CHAR、VARCHAR、LONG VARCHAR、CLOB、CLOB_FILE 或 CLOB_LOCATION_SPECIFIER 列。如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。
DATE	创建与数据库列完全相同的 DATE 列。	类型为 DATE 的 PC/IXF 列是常规输入。import 实用程序还会尝试接受任何字符类型的列，但长度不兼容的列除外。PC/IXF 文件中的字符列必须包含格式与目标数据库的地域代码一致的日期。
DBCLOB	创建 PC/IXF DBCLOB 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	如果 PC/IXF 列双字节代码页值等于数据库列的对应值，那么可以接受 PC/IXF GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC、DBCLOB、DBCLOB_FILE 或 DBCLOB_LOCATION_SPECIFIER 列。如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。
DECIMAL	创建与数据库列完全相同的 DECIMAL 列。该列的精度和标度存储在列描述符记录中。	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的列。如果各个值不在要导入至其中的 DECIMAL 列的范围内，那么会拒绝这些值。
DECFLOAT	创建与数据库列完全相同的 DECFLOAT 列。该列的精度存储在列描述符记录中。	接受下列类型的列：SMALLINT、INTEGER、BIGINT（仅转换为 DECFLOAT(34)）、DECIMAL、FLOAT、REAL、DOUBLE 或 DECFLOAT(16)（仅转换为 DECFLOAT(34)）。其他数字列类型对于 DECFLOAT 有效，但如果值不符合目标精度，将会进行舍入。
FLOAT	创建与数据库列完全相同的 FLOAT 列。	接受任何数字类型（SMALLINT、INTEGER、BIGINT、DECIMAL 或 FLOAT）的列。所有值都在范围内。
GRAPHIC（仅适用于 DBCS）	创建 PC/IXF GRAPHIC 列。数据库列长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	如果 PC/IXF 列双字节代码页值等于数据库列的对应值，那么可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC 列。如果 PC/IXF 列为固定长度，那么其长度必须与数据库列长度相兼容。必要时将在数据的右边填充双字节空格（x'8140'）。

表 54. PC/IXF 文件格式可接受的数据类型格式 (续)

数据类型	**export 实用程序创建的文件中的格式**	**import 实用程序可接受的格式**
INTEGER	创建与数据库列完全相同的 INTEGER 列。	接受任何数字类型（SMALLINT、INTEGER、BIGINT、 DECIMAL 或 FLOAT）的列。如果各个值不在范围 -2 147 483 648 到 2 147 483 647 之间，那么会拒绝这些值。
LONG VARCHAR	创建 PC/IXF LONG VARCHAR 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	满足下列条件时，可接受 PC/IXF CHAR、VARCHAR 或 LONG VARCHAR 列: • 数据库列被标记为“用于位数据” • PC/IXF 列单字节代码页值等于数据库列的 SBCS CPGID，并且 PC/IXF 列双字节代码页值等于零或数据库列的 DBCS CPGID。 如果数据库列标记为“用于位数据”，那么还可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC 列。在任何情况下，如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。
LONG VARGRAPHIC（仅适用于 DBCS）	创建 PC/IXF LONG VARGRAPHIC 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	如果 PC/IXF 列双字节代码页值等于数据库列的对应值，那么可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC 列。如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。
SMALLINT	创建与数据库列完全相同的 SMALLINT 列。	接受任何数字类型（SMALLINT、INTEGER、BIGINT、 DECIMAL 或 FLOAT）的列。如果各个值不在范围 -32 768 到 32 767 之间，那么会拒绝这些值。
TIME	创建与数据库列完全相同的 TIME 列。	类型为 TIME 的 PC/IXF 列是常规输入。import 实用程序还会尝试接受任何字符类型的列，但长度不兼容的列除外。PC/IXF 文件中的字符列必须包含格式与目标数据库的地域代码一致的时间数据。
TIMESTAMP	创建与数据库列完全相同的 TIMESTAMP 列。	类型为 TIMESTAMP 的 PC/IXF 列是常规输入。import 实用程序还会尝试接受任何字符类型的列，但长度不兼容的列除外。PC/IXF 文件中的字符列必须包含使用时间戳记输入格式的数据。

表 54. PC/IXF 文件格式可接受的数据类型格式 (续)

数据类型	export 实用程序创建的文件中的格式	import 实用程序可接受的格式
VARCHAR	如果数据库列的最大长度为 254，那么创建 PC/IXF VARCHAR 列。如果数据库列的最大长度大于 254，那么创建 PC/IXF LONG VARCHAR 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	满足下列条件时，可接受 PC/IXF CHAR、VARCHAR 或 LONG VARCHAR 列: • 数据库列被标记为“用于位数据” • PC/IXF 列单字节代码页值等于数据库列的 SBCS CPGID，并且 PC/IXF 列双字节代码页值等于零或数据库列的 DBCS CPGID。 如果数据库列标记为“用于位数据”，那么还可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC 列。在任何情况下，如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。
VARGRAPHIC（仅适用于 DBCS）	如果数据库列的最大长度为 127，那么创建 PC/IXF VARGRAPHIC 列。如果数据库列的最大长度大于 127，那么创建 PC/IXF LONG VARGRAPHIC 列。数据库列的最大长度、SBCS CPGID 值和 DBCS CPGID 值将复制至列描述符记录。	如果 PC/IXF 列双字节代码页值等于数据库列的对应值，那么可以接受 PC/IXF GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC 列。如果 PC/IXF 列为固定长度，那么其长度必须与数据库列的最大长度相兼容。

控制 PC/IXF 文件导入数据库的一般规则

在 SBCS 或 DBCS 环境中导入 PC/IXF 文件时，数据库管理器 import 实用程序将应用下列一般规则:

- import 实用程序仅接受 PC/IXF 格式文件（IXFHID = 'IXF'）。不能导入其他格式的 IXF 文件。
- import 实用程序拒绝超过 1024 列的 PC/IXF 文件。
- 导出至 IXF 格式时，如果标识超出 IXF 格式支持的最大大小，那么导出操作会成功，但使用 CREATE 方式的后续导入操作不能使用生成的数据文件。将返回 SQL27984W。

  注: 不推荐使用 IMPORT 命令的 CREATE 和 REPLACE_CREATE 选项，将来的发行版中可能会除去这两个选项。
- PC/IXF H 记录中的 IXFHSBCP 的值必须等于 SBCS CPGID，或者 IXFHSBCP/IXFHDBCP 与目标数据库的 SBCS/DBCS CPGID 之间必须存在转换表。IXFHDBCP 的值必须等于“00000”或目标数据库的 DBCS CPGID。如果不满足其中任一条件，import 实用程序都会拒绝 PC/IXF 文件，除非指定 FORCEIN 选项。
- 无效数据类型 – 新表

将 PC/IXF 文件导入至*新*表这一过程是由 IMPORT 命令中的 CREATE 或 REPLACE_CREATE 关键字指定的。如果对导入至新表这一过程指定了数据类型无效的 PC/IXF 列，那么 import 实用程序将终止。将拒绝整个 PC/IXF 文件，不创建任何表，也不会导入任何数据。

- 无效数据类型 – 现有表

  将 PC/IXF 文件导入到*现有*表这一过程是由 IMPORT 命令中的 INSERT、INSERT_UPDATE、REPLACE 或 REPLACE_CREATE 关键字指定的。如果对导入至现有表这一过程指定了数据类型无效的 PC/IXF 列，那么可能进行下列两个操作中的一个:

  – 如果目标表列可空，那么会忽略无效 PC/IXF 列的所有值，并且表列值将设置为 NULL。
  – 如果目标表列不可空，那么 import 实用程序将终止。将拒绝整个 PC/IXF 文件，也不会导入任何数据。现有表将保持不变。

- 导入到新表中时，可空 PC/IXF 列将生成可空数据库列，而不可空 PC/IXF 列将生成不可空数据库列。
- 可将不可空 PC/IXF 列导入到可空数据库列中。
- 可将可空 PC/IXF 列导入到不可空数据库列中。如果在 PC/IXF 列中遇到 NULL 值，那么 import 实用程序将拒绝包含 NULL 值的 PC/IXF 行中的所有列值（拒绝整行），并继续处理下一个 PC/IXF 行。即，如果目标表列（对于 NULL）不可空，那么不会从包含 NULL 值的 PC/IXF 行导入数据。
- 不兼容列 – 新表

  在导入至*新*数据库表期间，如果选择了与目标数据库列不兼容的 PC/IXF 列，那么 import 实用程序将终止。将拒绝整个 PC/IXF 文件，不创建任何表，也不会导入任何数据。

  注: IMPORT 的 FORCEIN 选项超出兼容列的作用域。

- 不兼容列 – 现有表

  在导入至*现有*数据库表期间，如果选择了与目标数据库列不兼容的 PC/IXF 列，那么可能采取下列两个操作中的一个:

  – 如果目标表列可空，那么会忽略 PC/IXF 列的所有值，并且表列值将设置为 NULL。
  – 如果目标表列不可空，那么 import 实用程序将终止。将拒绝整个 PC/IXF 文件，也不会导入任何数据。现有表将保持不变。

  注: IMPORT 的 FORCEIN 选项超出兼容列的作用域。

- 无效值

  在导入期间，如果遇到对目标数据库列无效的 PC/IXF 列值，那么 import 实用程序将拒绝包含无效值的 PC/IXF 行中的所有列值（拒绝整行），并继续处理下一个 PC/IXF 行。

控制 PC/IXF 文件导入数据库的特定于数据类型的规则

- 可将有效 PC/IXF 数字列导入至任何兼容数据数据库列。因为包含 4 字节浮点数据的 PC/IXF 列不是有效数据类型，所以不会导入这些列。

- 数据库日期/时间列可接受匹配的 PC/IXF 日期/时间列（DATE、TIME 和 TIMESTAMP）中的值，也可以接受 PC/IXF 字符列（CHAR、VARCHAR 和 LONG VARCHAR）中的值，同时要受列长度和值兼容性限制的约束。
- 总是可将有效的 PC/IXF 字符列（CHAR、VARCHAR 或 LONG VARCHAR）导入至标记为“用于位数据”的 *现有*数据库字符列，或者:
  - IXFCSBCP 和 SBCS CPGID 必须一致
  - 必须有用于 IXFCSBCP/IXFCDBCP 和 SBCS/DBCS 的转换表
  - 一个集合必须全部设置为零（用于位数据）。

  如果 IXFCSBCP 不为零，那么 IXFCDBCP 的值必须等于零或目标数据库列的 DBCS CPGID。

  如果不满足其中任一条件，PC/IXF 和数据库列就不能兼容。

  将有效 PC/IXF 字符列导入到*新*数据库表中时，IXFCSBCP 的值必须等于零或数据库的 SBCS CPGID，或者必须存在转换表。如果 IXFCSBCP 为零，那么 IXFCDBCP 必须也为零（否则 PC/IXF 列是无效数据类型）；IMPORT 在新表中创建标记为“用于位数据”的字符列。如果 IXFCSBCP 不为零，并且等于数据库的 SBCS CPGID，那么 IXFCDBCP 的值必须等于零或数据库的 DBCS CPGID；在此情况下，该实用程序在新表中创建 SBCS 和 DBCS CPGID 值等于数据库中的对应值的字符列。如果不满足其中任一条件，PC/IXF 和数据库列就不能兼容。

  FORCEIN 选项可用来覆盖代码页相等性检查。但是，IXFCSBCP 等于零而 IXFCDBCP 不等于零的 PC/IXF 字符列是无效数据类型，因此不能导入，即使指定 FORCEIN 也一样。
- 总是可将有效 PC/IXF 图形列（GRAPHIC、VARGRAPHIC 或 LONG VARGRAPHIC）导入至标记为“用于位数据” 的*现有*数据库字符列，但它与所有其他数据库列不兼容。FORCEIN 选项可用来放松此限制。但是，IXFCSBCP 不等于零或 IXFCDBCP 等于零的 PC/IXF 图形列是无效数据类型，因此不能导入，即使指定 FORCEIN 也一样。

  将有效 PC/IXF 图形列导入至数据库图形列时，IXFCDBCP 的值必须等于目标数据库列的 DBCS CPGID（即两个列的双字节代码页必须一致）。
- 在将 PC/IXF 文件导入到现有数据库表中时，如果选择了长度超过目标列最大长度的定长字符串列（CHAR 或 GRAPHIC），那么这些列将会不兼容。
- 在将 PC/IXF 文件导入到现有数据库表中时，如果选择了长度超过目标列最大长度的变长字符串列（VARCHAR、LONG VARCHAR、VARGRAPHIC 或 LONG VARGRAPHIC），那么这些列将会兼容。各个值将按照控制 数据库管理器 INSERT 语句的兼容性规则进行处理，对于目标数据库列而言太长的 PC/IXF 值无效。
- 对于导入至定长数据库字符列（即 CHAR 列）的 PC/IXF 值，必要时将在其右边填充单字节空格（0x20）以获取长度等于数据库列长度的值。对于导入至定长数据库*图形*列（即 GRAPHIC 列）的 PC/IXF 值，必要时将在其右边填充双字节空格（0x8140）以获取长度等于数据库列长度的值。
- 因为 PC/IXF VARCHAR 列的最大长度为 254 个字节，所以最大长度为 *n*（254 *n* 4001）的数据库 VARCHAR 列必须导出至最大长度为 *n* 的 PC/IXF LONG VARCHAR 列。

- 尽管 PC/IXF LONG VARCHAR 列的最大长度为 32767 个字节，并且数据库 LONG VARCHAR 列的最大长度限制为 32700 个字节，但长度大于 32700 字节（但小于 32768 个字节）的 PC/IXF LONG VARCHAR 仍然有效，并且可导入至数据库 LONG VARCHAR 列，不过数据可能会丢失。
- 因为 PC/IXF VARGRAPHIC 列的最大长度为 127 个字节，所以最大长度为 *n*（127 *n* 2001）的数据库 VARGRAPHIC 列必须导出至最大长度为 *n* 的 PC/IXF LONG VARGRAPHIC 列。
- 尽管 PC/IXF LONG VARGRAPHIC 列的最大长度为 16383 个字节，并且数据库 LONG VARGRAPHIC 列的最大长度限制为 16350 个字节，但长度大于 16350 字节（但小于 16384 个字节）的 PC/IXF LONG VARGRAPHIC 仍然有效，并且可导入至数据库 LONG VARGRAPHIC 列，不过数据可能会丢失。

表 55 和 表 56 总结了在不使用 FORCEIN 选项的情况下将 PC/IXF 文件导入至新的或现有数据库表的过程。

表 55. 不使用 FORCEIN 选项的 PC/IXF 文件导入总结 - 数字类型

	DATABASE COLUMN DATA TYPE					
PC/IXF COLUMN DATA TYPE	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**DFP**	**FLT**
-SMALLINT	N					
	E	E	E	E[a]	E	E
-INTEGER		N				
	E[a]	E	E	E[a]	E	E
-BIGINT			N			
	E[a]	E[a]	E	E[a]	E	E
-DECIMAL				N		
	E[a]	E[a]	E[a]	E[a]	E	E
-DECFLOAT						
	E[a]	E[a]	E[a]	E[a]	E	E[a]
-FLOAT						N
	E[a]	E[a]	E[a]	E[a]	E	E

[a] 如果各个值在目标数字数据类型的范围之外，那么会拒绝这些值。

表 56. 不使用 FORCEIN 选项的 PC/IXF 文件导入总结 - 字符、图形和日期/时间类型

	DATABASE COLUMN DATA TYPE						
PC/IXF COLUMN DATA TYPE	**(0,0)**	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAP H[b]**	**DATE**	**TIME**	**TIME STAMP**
-(0,0)	N						
	E				E[c]	E[c]	E[c]
-(SBCS,0)		N	N				
	E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)			N		E[c]	E[c]	E[c]
	E		E				
-GRAPHIC				N			

表 56. 不使用 FORCEIN 选项的 PC/IXF 文件导入总结 - 字符、图形和日期/时间类型 (续)

	DATABASE COLUMN DATA TYPE						
PC/IXF COLUMN DATA TYPE	**(0,0)**	**(SBCS, 0)[d]**	**(SBCS, DBCS)[b]**	**GRAP H[b]**	**DATE**	**TIME**	**TIME STAMP**
	E			E			
-DATE					N		
					E		
-TIME						N	
						E	
-TIME STAMP							N
							E

[b] 数据类型仅在 DBCS 环境中可用。

[c] 如果各个值不是有效的日期或时间值，那么会拒绝这些值。

[d] 数据类型在 DBCS 环境中不可用。

注:

1. 该表是所有有效 PC/IXF 和数据库管理器数据类型的矩阵。如果可将 PC/IXF 列导入至数据库列，那么将在矩阵单元中显示一个字母，该字母位于 PC/IXF 数据类型矩阵行与数据库管理器数据类型矩阵列的交点处。“N”指示实用程序正在创建新数据库表（将创建指示的数据类型的数据库列）。“E”指示实用程序正在将数据导入至现有数据库表（指示的数据类型的数据库列是有效目标）。
2. 字符串数据类型可通过代码页属性来区分。这些属性显示为有序对 (SBCS,DBCS)，其中:
   - SBCS 为零值或指示字符数据类型的单字节代码页属性的非零值。
   - DBCS 为零值或指示字符数据类型的双字节代码页属性的非零值。
3. 如果该表指示可将 PC/IXF 字符列导入至数据库字符列，那么其相应代码页属性对的值将符合用于管理代码页相等性的规则。

PC/IXF 与 System/370 IXF V0 之间的差别

下面描述数据库管理器使用的 PC/IXF 与一些主机数据库产品使用的 System/370 IXF 之间的差别:

- PC/IXF 文件定向于 ASCII 而不是 EBCDIC。PC/IXF 文件大幅扩展了代码页标识，包括 H 记录中的新代码页标识以及列描述符记录中的实际代码页值使用。它还提供了用于将字符数据列标记为“用于位数据”的机制。“用于位数据”列具有特殊意义，这是因为 PC/IXF 文件格式与任何其他 IXF 或数据库文件格式之间的变换不能对“用于位数据”列中包含的值执行代码页转换。
- 只允许机器数据格式；即 IXFTFORM 字段必须始终包含值 M。而且机器必须为 PC 格式；即 IXFTMFRM 字段必须包含值 PC。这表示 PC/IXF 数据记录的数据部分中的整数、浮点数和十进制数必须为 PC 格式。
- 在 PC/IXF 文件中的 H 记录之后的任何位置允许出现应用程序（A）记录。在计算 IXFHHCNT 字段的值时，不会对它们进行计数。

- 每个 PC/IXF 记录都以记录长度指示符开头。这是一个由 6 个字节组成的字符，表示包含 PC/IXF 记录长度的整数值（以字节计），它不包括记录长度指示符本身，即总记录长度减去 6 个字节。记录长度字段的用途是允许 PC 程序标识记录边界。
- 为便于压缩变长数据的存储空间，并且避免在字段分割为多个记录时的复杂处理，PC/IXF 不支持版本 0 IXF X 记录，而是支持 D 记录标识。每当变长字段或可空字段是数据 D 记录中的最后一个字段时，不必将字段的完整最大长度写至 PC/IXF 文件。

FORCEIN 选项

forcein 文件类型修饰符允许导入 PC/IXF 文件，而不理会 PC/IXF 文件与目标数据库中的数据之间的代码页差别。它允许更加灵活地定义兼容列。

forcein 的常规语义

在 SBCS 或 DBCS 环境中使用 forcein 文件类型修饰符时，下列一般语义适用:

- 使用 forcein 文件类型修饰符时应特别谨慎。通常建议在未启用此选项的情况下尝试导入。但是，因为 PC/IXF 数据交换体系结构的一般特征，某些 PC/IXF 文件可能包含导入时必须进行干预的数据类型和值。
- 如果使用 forcein 导入至*新*表，那么可能会产生与导入至现有表不同的结果。现有表对每个 PC/IXF 数据类型预定义了目标数据类型。
- 如果使用 lobsinfile 文件类型修饰符导出 LOB 数据，并且文件移至具有不同代码页的另一客户机，那么在导入或装入至数据库时，位于单独文件中的 CLOBS 和 DBCLOBS 不会转换至客户机代码页，这与其他数据是不同的。

forcein 的代码页语义

在 SBCS 或 DBCS 环境中使用 forcein 文件类型修饰符时，下列代码页语义适用:

- forcein 文件类型修饰符禁用所有 import 实用程序代码页比较。

  在导入至新的或现有的数据库表时，此规则适用于列级别和文件级别的代码页比较。在列（如数据类型）级别，此规则仅适用于下列数据库管理器和 PC/IXF 数据类型：字符（CHAR、VARCHAR 和 LONG VARCHAR）以及图形（GRAPHIC、VARGRAPHIC 和 LONG VARGRAPHIC）。因为其他数据类型的代码页属性与数据类型值的解释无关，所以存在限制。
- forcein 不会禁用为确定数据类型而进行的代码页属性检查。

  例如，数据库管理器允许使用“用于位数据”属性来声明 CHAR 列。这种声明将该列的 SBCS CPGID 和 DBCS CPGID 设置为零；这是将列值标识为二进制位串（而不是字符串）的 CPGID 的零值。
- forcein 未暗示代码页转换。

  与 forcein 文件类型修饰符有关的文件类型的值将按原样复制。不会使用代码点映射来进行代码页环境更改。对于定长目标列，用空格填充导入的值可能是必需的。
- 使用 forcein 将数据导入至*现有*表时:
  - 目标数据库表和列的代码页值总是处于优先位置。
  - 将忽略 PC/IXF 文件和列的代码页值。

无论是否使用 forcein 选项，此规则都适用。一旦创建数据库，数据库管理器就不允许更改数据库或列代码页值。

- 使用 forcein 导入至*新*表时:
  - 目标数据库的代码页值将处于优先位置。
  - 带有 IXFCSBCP = IXFCDBCP = 0 的 PC/IXF 字符列将生成标记为“用于位数据”的表列。
  - 所有其他 PC/IXF 字符列将生成 SBCS 和 DBCS CPGID 值等于数据库的相应值的表字符列。
  - PC/IXF 图形列生成具有“未定义”的 SBCS CPGID 的表图形列，而 DBCS CPGID 等于数据库的相应值（仅适用于 DBCS 环境）。

forcein 示例

考虑 IXFCSBCP = ’00897’ 且 IXFCDBCP = ’00301’ 的 PC/IXF CHAR 列。此列将导入至 SBCS CPGID = ’00850’ 且 DBCS CPGID = ’00000’ 的数据库 CHAR 列。如果不使用 forcein，那么该实用程序终止，并且不会导入任何数据，或者将忽略 PC/IXF 列值，并且数据库列包含 NULL（如果数据库列可空）。使用 forcein 时，该实用程序将继续进行并忽略代码页不兼容问题。如果不存在其他数据类型不兼容问题（如长度），那么 PC/IXF 列的值将按原样导入，并且在数据库列代码页环境下可供解释之用。

下面两个表显示:

- 当导入带有指定代码页属性的 PC/IXF 文件数据类型时，将在*新*数据库表中创建列代码页属性。
- 如果 PC/IXF 数据类型无效或不兼容，那么 import 实用程序将拒绝这些数据类型。

表 57. 用于 SBCS 的 import 实用程序代码页语义（新表）总结. 此表假定 a 与 x 之间不存在转换表。如果存在，那么第 3 项和第 4 项将在不使用 forcein 的情况下成功运行。

PC/IXF 数据类型的代码页属性	数据库表列的代码页属性	
	不使用 forcein	使用 forcein
(0,0)	(0,0)	(0,0)
(a,0)	(a,0)	(a,0)
(x,0)	拒绝	(a,0)
(x,y)	拒绝	(a,0)
(a,y)	拒绝	(a,0)
(0,y)	拒绝	(0,0)
注: 1. 请参阅表 58 的注释。		

表 58. 用于 DBCS 的 import 实用程序代码页语义（新表）总结. 此表假定 a 与 x 之间不存在转换表。

PC/IXF 数据类型的代码页属性	数据库表列的代码页属性	
	不使用 forcein	使用 forcein
(0,0)	(0,0)	(0,0)
(a,0)	(a,b)	(a,b)
(x,0)	拒绝	(a,b)

表 58. 用于 DBCS 的 import 实用程序代码页语义（新表）总结 (续). 此表假定 a 与 x 之间不存在转换表。

PC/IXF 数据类型的代码页属性	数据库表列的代码页属性	
	不使用 forcein	使用 forcein
(a,b)	(a,b)	(a,b)
(x,y)	拒绝	(a,b)
(a,y)	拒绝	(a,b)
(x,b)	拒绝	(a,b)
(0,b)	(-,b)	(-,b)
(0,y)	拒绝	(-,b)

注:

1. PC/IXF 数据类型的代码页属性将显示为有序对，其中 x 表示非零单字节代码页值，而 y 表示非零双字节代码页值。'-' 表示未定义代码页值。
2. 在各种代码页属性对中使用不同字母时请谨慎进行。不同的字母代表不同的值。例如，如果 PC/IXF 数据类型显示为 (x,y)，而数据库列显示为 (a,y)，那么 x 不等于 a，但 PC/IXF 文件和数据库具有相同的双字节代码页值 y。
3. 只有字符和图形数据类型才会受 forcein 代码页语义影响。
4. 假定包含新表的数据库的代码页属性为 (a,0)；因此新表中的所有字符列的代码页属性都必须为 (0,0) 或 (a,0)。

   在 DBCS 环境中，假定包含新表的数据库的代码页属性为 (a,b)；因此，新表中的所有图形列的代码页属性都必须为 (-,b)，并且所有字符列的代码页属性都必须为 (a,b)。原因是未对图形数据类型定义 SBCS CPGID，所以它显示为 '-'。
5. 结果的数据类型由 forcein 数据类型语义中描述的规则确定。
6. 拒绝结果是规则对无效或不兼容数据类型的反应。

下面两个表显示:

- import 实用程序接受将带有各种代码页属性的 PC/IXF 数据类型导入至具有指定代码页属性的*现有*表列（*目标*列）。
- import 实用程序接受不允许将带有特定代码页属性的 PC/IXF 数据类型导入至具有显示的代码页属性的*现有*表列。如果 PC/IXF 数据类型无效或不兼容，那么实用程序将拒绝这些数据类型。

表 59. 用于 SBCS 的 import 实用程序代码页语义（现有表）总结. 此表假定 a 与 x 之间不存在转换表。

PC/IXF 数据类型的代码页属性	目标数据库列的代码页属性	导入结果	
		不使用 forcein	使用 forcein
(0,0)	(0,0)	接受	接受
(a,0)	(0,0)	接受	接受
(x,0)	(0,0)	接受	接受
(x,y)	(0,0)	接受	接受
(a,y)	(0,0)	接受	接受
(0,y)	(0,0)	接受	接受
(0,0)	(a,0)	空或拒绝	接受

表 59. 用于 SBCS 的 import 实用程序代码页语义（现有表）总结 (续). 此表假定 a 与 x 之间不存在转换表。

PC/IXF 数据类型的代码页属性	目标数据库列的代码页属性	导入结果	
		不使用 forcein	使用 forcein
(a,0)	(a,0)	接受	接受
(x,0)	(a,0)	空或拒绝	接受
(x,y)	(a,0)	空或拒绝	接受
(a,y)	(a,0)	空或拒绝	接受
(0,y)	(a,0)	空或拒绝	空或拒绝
注: 1. 请参阅第 382 页的表 57 的注释。 2. 空或拒绝结果是规则对无效或不兼容数据类型的反应。			

表 60. 用于 DBCS 的 import 实用程序代码页语义（现有表）总结. 此表假定 a 与 x 之间不存在转换表。

PC/IXF 数据类型的代码页属性	目标数据库列的代码页属性	导入结果	
		不使用 forcein	使用 forcein
(0,0)	(0,0)	接受	接受
(a,0)	(0,0)	接受	接受
(x,0)	(0,0)	接受	接受
(a,b)	(0,0)	接受	接受
(x,y)	(0,0)	接受	接受
(a,y)	(0,0)	接受	接受
(x,b)	(0,0)	接受	接受
(0,b)	(0,0)	接受	接受
(0,y)	(0,0)	接受	接受
(0,0)	(a,b)	空或拒绝	接受
(a,0)	(a,b)	接受	接受
(x,0)	(a,b)	空或拒绝	接受
(a,b)	(a,b)	接受	接受
(x,y)	(a,b)	空或拒绝	接受
(a,y)	(a,b)	空或拒绝	接受
(x,b)	(a,b)	空或拒绝	接受
(0,b)	(a,b)	空或拒绝	空或拒绝
(0,y)	(a,b)	空或拒绝	空或拒绝
(0,0)	(-,b)	空或拒绝	接受
(a,0)	(-,b)	空或拒绝	空或拒绝
(x,0)	(-,b)	空或拒绝	空或拒绝
(a,b)	(-,b)	空或拒绝	空或拒绝
(x,y)	(-,b)	空或拒绝	空或拒绝

表 60. 用于 DBCS 的 import 实用程序代码页语义（现有表）总结 (续). 此表假定 a 与 x 之间不存在转换表。

PC/IXF 数据类型的代码页属性	目标数据库列的代码页属性	导入结果	
		不使用 forcein	使用 forcein
(a,y)	(-,b)	空或拒绝	空或拒绝
(x,b)	(-,b)	空或拒绝	空或拒绝
(0,b)	(-,b)	接受	接受
(0,y)	(-,b)	空或拒绝	接受

注:

1. 请参阅第 382 页的表 57 的注释。
2. 空或拒绝结果是规则对无效或不兼容数据类型的反应。

forcein 的数据类型语义

forcein 文件类型修饰符允许将特定 PC/IXF 列导入至不相等并且本来具有不兼容数据类型的目标数据库列。在 SBCS 或 DBCS 环境中使用 forcein 时（特别说明时除外），下列数据类型语义适用:

- 在 SBCS 环境中，forcein 允许:
  - 将 PC/IXF BIT 数据类型（对于 PC/IXF 字符列，IXFCSBCP = 0 = IXFCDBCP）导入至数据库字符列（非零 SBCS CPGID 并且 DBCS CPGID = 0）；仅适用于现有表
  - 将 PC/IXF MIXED 数据类型（非零 IXFCSBCP 和 IXFCDBCP）导入至数据库字符列；同时适用于新表和现有表
  - 将 PC/IXF GRAPHIC 数据类型导入至数据库“用于位数据”列（SBCS CPGID = 0 = DBCS CPGID）；仅适用于新表（总是允许对现有表执行此操作）。
- forcein 文件类型修饰符不会扩展有效 PC/IXF 数据类型的作用域。

  无论是否使用 forcein，数据类型未定义为有效 PC/IXF 数据类型的 PC/IXF 列对导入操作都是无效的。
- 在 DBCS 环境中，forcein 允许:
  - 将 PC/IXF BIT 数据类型导入至数据库字符列
  - 将 PC/IXF BIT 数据类型导入至数据库图形列；但是，如果 PC/IXF BIT 列是定长，那么该长度必须为偶数。如果定长 PC/IXF BIT 列为奇数长度，那么该列与数据库图形列不兼容。无论长度是奇数还是偶数，变长 PC/IXF BIT 列都是兼容的，即使变长列的奇数长度值对导入至数据库图形列操作而言是无效值。
  - 将 PC/IXF MIXED 数据类型导入至数据库字符列。

第 386 页的表 61 总结了在指定了 forcein 的情况下将 PC/IXF 文件导入至新的或现有数据库表的过程。

表 61. 使用 forcein 的 PC/IXF 文件导入总结

	DATABASE COLUMN DATA TYPE											
PC/IXF COLUMN DATA TYPE	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**
-SMALLINT	N											
	E	E	E	E[a]	E							
-INTEGER		N										
	E[a]	E	E	E[a]	E							
-BIGINT			N									
	E[a]	E[a]	E	E[a]	E							
-DECIMAL				N								
	E[a]	E[a]	E[a]	E[a]	E							
-FLOAT					N							
	E[a]	E[a]	E[a]	E[a]	E							
-(0,0)						N						
						E	E w/F	E w/F	E w/F	E[c]	E[c]	E[c]
-(SBCS,0)							N	N				
						E	E	E		E[c]	E[c]	E[c]
-(SBCS, DBCS)							N w/F[d]	N		E[c]	E[c]	E[c]
						E	E w/F	E				
-GRAPHIC						N w/F[d]			N			
						E			E			
-DATE										N		
										E		
-TIME											N	
											E	
-TIME STAMP												N
												E

表 61. 使用 forcein 的 PC/IXF 文件导入总结 (续)

	DATABASE COLUMN DATA TYPE											
PC/IXF COLUMN DATA TYPE	**SMALL INT**	**INT**	**BIGINT**	**DEC**	**FLT**	**(0,0)**	**(SBCS, 0)[e]**	**(SBCS, DBCS)[b]**	**GRAPH[b]**	**DATE**	**TIME**	**TIME STAMP**

注: 如果只有在使用 forcein 时才能将 PC/IXF 列导入至数据库列，那么字符串"w/F"将与"N"或"E"一起显示。"N"指示实用程序正在创建新数据库表；"E"指示实用程序正在将数据导入至现有数据库表。forcein 文件类型修饰符只影响字符和图形数据类型的兼容性。

[a] 如果各个值在目标数字数据类型的范围之外，那么会拒绝这些值。

[b] 数据类型仅在 DBCS 环境中可用。

[c] 如果各个值不是有效的日期或时间值，那么会拒绝这些值。

[d] 仅当目标数据库不支持源 PC/IXF 数据类型时适用。

[e] 数据类型在 DBCS 环境中不可用。

工作表文件格式（WSF）

Lotus 1-2-3 和 Symphony 产品使用同一基本格式，并且在每个发行版添加了附加功能。数据库管理器支持一小部分工作表记录，这些记录对于所有 Lotus 产品来说都是相同的。即，对于数据库管理器支持的 Lotus 1-2-3 和 Symphony 产品发行版来说，接受所有带有任何三字符扩展名的文件；例如: WKS、WK1、WRK、WR1 和 WJ2。

每个 WSF 文件都代表一个工作表。数据库管理器使用下列约定来解释工作表以及使其导出操作生成的工作表保持一致:

- 第一行（ROW 值为 0）中的单元格是为关于整个工作表的描述性信息保留的。此行中的所有数据都是可选的。在导入期间将忽略这些数据。
- 第二行（ROW 值为 1）中的单元格用于列标号。
- 其余各行是数据行（记录，或表中的数据行）。
- 任何列标题下的单元格值都是该特定列或字段的值。
- 如果在一行单元格内容记录的特定列中不存在实际的单元格内容记录（例如，没有整数、数字、标签或公式记录），那么指示值为 NULL。

  注: 既不会导入也不会导出包含 NULL 的行。

要在导出操作期间创建与 WSF 格式相符的文件，可能会丢失一些数据。

WSF 文件使用 Lotus 代码点映射，该代码点映射不一定与 DB2 数据库支持的现有代码页相同。因此，在导入或导出 WSF 文件时，将在 Lotus 代码点与应用程序代码页使用的代码点之间进行数据转换。DB2 支持在 Lotus 代码点与代码页 437、819、850、860、863 和 865 定义的代码点之间进行转换。

注: 对于多字节字符集用户来说，不会执行转换。

数据移动的 Unicode 注意事项

连接到非 Unicode 数据库的 Unicode 客户机不支持使用 EXPORT、IMPORT 和 LOAD 实用程序。

如本节所述，Unicode 数据库支持 DEL、ASC 和 PC/IXF 文件格式。不支持 WSF 格式。

当从 Unicode 数据库导出数据到 ASCII 定界（DEL）文件时，将把所有字符数据转换到应用程序代码页，即将字符字符串数据和图形字符串数据都转换到客户机的同一 SBCS 或 MBCS 代码页。在导出任何数据库时，这是预期的行为。由于整个定界 ASCII 文件只能使用一个代码页，所以不能更改此行为。因此，如果导出到定界 ASCII 文件，就会只保存应用程序代码页中存在的那些 UCS-2 字符，而将其他字符替换为应用程序代码页中的缺省替换字符。对于 UTF-8 客户机（代码页 1208）来说，由于 UTF-8 客户机支持所有 UCS-2 字符，所以不会丢失数据。

当从 ASCII 文件（DEL 或 ASC）导入到 Unicode 数据库时，将把字符串数据从应用程序代码页转换为 UTF-8，并将图形字符串数据从应用程序代码页转换为 UCS-2。不会丢失数据。如果要导入使用另一代码页保存的 ASCII 数据，那么应该在发出 IMPORT 命令前更改数据文件代码页。可以通过将 **DB2CODEPAGE** 注册表变量设置为 ASCII 数据文件的代码页或使用 `codepage` 文件类型修饰符来指定数据文件的代码页。

SBCS 和 MBCS 客户机的有效 ASCII 定界符范围与 IBM DB2 V9.1 对那些客户机支持的 ASCII 定界符范围完全相同。UTF-8 客户机的有效定界符范围是 X'01' 到 X'7F'，并且惯例限制适用。

当从 Unicode 数据库导出到 PC/IXF 文件时，将把字符字符串数据转换为客户机的 SBCS/MBCS 代码页。不转换图形字符串数据，这些数据将使用 UCS-2（代码页 1200）存储。不会丢失数据。

当从 PC/IXF 文件导入到 Unicode 数据库时，会假定字符字符串数据使用 PC/IXF 头中存储的 SBCS/MBCS 代码页，并假定图形字符串数据使用 PC/IXF 头中存储的 DBCS 代码页。IMPORT 实用程序将字符字符串数据从 PC/IXF 头中指定的代码页转换为客户机代码页，然后从客户机代码页转换为 UTF-8（由 INSERT 语句转换）。IMPORT 实用程序将图形字符串数据从 PC/IXF 头中指定的 DBCS 代码页直接转换为 UCS-2（代码页 1200）。

LOAD 实用程序将数据直接放入数据库，缺省情况下，假定 ASC 或 DEL 文件中的数据使用数据库的代码页。因此，缺省情况下，对 ASCII 文件不执行代码页转换操作。当使用 `codepage` 修饰符显式指定了数据文件的代码页时，LOAD 实用程序将使用此信息来从指定的代码页转换为数据库代码页，然后再装入数据。对于 PC/IXF 文件，LOAD 实用程序始终从 IXF 头中指定的代码页转换为数据库代码页（对于 CHAR 来说是 1208，对于 GRAPHIC 来说是 1200）。

DBCLOB 文件的代码页始终是 1200（即 UCS-2）。CLOB 文件的代码页与所导入、装入或导出的数据文件的代码页相同。例如，在使用 PC/IXF 格式装入或导入数据时，假定 CLOB 文件使用 PC/IXF 头指定的代码页。如果 DBCLOB 文件是 ASC 或 DEL 格式的，那么 LOAD 实用程序会假定 CLOB 数据使用数据库的代码页，而 IMPORT 实用程序假定该数据使用客户机应用程序的代码页。

始终对 Unicode 数据库指定 nochecklengths 修饰符，原因如下:

- 可以将任何 SBCS 客户机连接到没有 DBCS 代码页的数据库
- 通常，UTF-8 格式的字符字符串与使用客户机代码页的那些字符字符串长度不同。

代码页 1394、1392 和 5488 的注意事项

可以使用 IMPORT、EXPORT 和 LOAD 实用程序来将中文代码页 GB 18030（代码页标识为 1392 和 5488）和日语代码页 ShiftJISX 0213（代码页标识为 1394）的数据传输到 DB2 Unicode 数据库中。此外，可使用 EXPORT 实用程序来将 DB2 Unicode 数据库中的数据传输到 GB 18030 或 ShiftJIS X0213 代码页。

例如，以下命令将把所连接的远程客户机上的 Shift_JISX0213 数据文件 u/jp/user/x0213/data.del 装入到 MYTABLE 中:

```
db2 load client from /u/jp/user/x0213/data.del
of del modified by codepage=1394 insert into mytable
```

其中，MYTABLE 在 DB2 Unicode 数据库中。

由于只支持在 Unicode 客户机与 Unicode 服务器之间建立连接，因此，在使用 LOAD、Import 或 EXPORT 实用程序之前，需要使用 Unicode 客户机或者将 DB2 注册表变量 **DB2CODEPAGE** 设置为 1208。

从代码页 1394、1392 或 5488 转换为 Unicode 会导致扩展。例如，可以将 2 字节字符作为两个 16 位 Unicode 字符存储在 GRAPHIC 列中。您需要确保 Unicode 数据库中的目标列宽度足以包含任何扩展的 Unicode 字节。

不兼容性

对于连接到 Unicode 数据库的应用程序来说，图形字符串数据始终使用 UCS-2（代码页 1200）。对于连接到非 Unicode 数据库的应用程序来说，图形字符串数据使用应用程序的 DBCS 代码页，或者，如果应用程序代码页是 SBCS，那么不允许使用图形字符串数据。例如，当 932 客户机连接到日语非 Unicode 数据库时，图形字符串数据使用代码页 301。对于连接到 Unicode 数据库的 932 客户机应用程序来说，图形字符串数据使用 UCS-2 编码。

字符集和本地语言支持

DB2 数据移动实用程序提供了以下本地语言支持（NLS）:

- import 实用程序和 export 实用程序提供从客户机代码页至服务器代码页的自动代码页转换。
- 对于 load 实用程序，可通过将 codepage 修饰符与 DEL 和 ASC 文件配合使用将数据从任意代码页转换为服务器代码页。
- 对于所有实用程序，IXF 数据将从其原始代码页（存储在 IXF 文件中）自动转换为服务器代码页。

有时，会出现代码页不等同的情况（扩展了或收缩了字符数据）。例如，日语或繁体中文扩展 UNIX 代码（EUC）和双字节字符集（DBCS）可能会将同一个字符编码成不同长度。通常，在读入任何数据之前比较输入数据的长度与目标列的长度。如果输入长度大于目标长度，并且该列可空，那么将把 NULL 插入到该列中。否则，将拒绝该请

求。如果指定了 nochecklengths 文件类型修饰符，那么不会执行初始比较，并且会尝试导入或装入数据。转换完成后，如果数据太长，就会拒绝该行。否则会导入或装入数据。

XML 数据移动

LOAD、IMPORT 和 EXPORT 实用程序提供了对 XML 数据移动的支持。

导入 XML 数据

可以使用 IMPORT 实用程序将 XML 文档插入到常规关系表中。只能导入结构良好的 XML 文档。

使用 IMPORT 命令的 XML FROM 选项指定要导入的 XML 文档的位置。XMLVALIDATE 选项指定验证已导入的文档的方式。可以选择通过以下方式验证已导入的 XML 数据: 针对用 IMPORT 命令指定的模式，针对源 XML 文档内的模式位置提示所标识的模式，或者通过主数据文件中的 XML 数据说明符所标识的模式。还可以使用 XMLPARSE 选项指定导入 XML 文档时处理空格的方式。xmlchar 和 xmlgraphic 文件类型修饰符允许您指定已导入的 XML 数据的编码特征。

装入 XML 数据

LOAD 实用程序提供了有效的方式将大量 XML 数据插入到表中。此实用程序还允许 IMPORT 实用程序未提供的特定选项，如从用户定义的游标导入的功能。

与 IMPORT 命令一样，可使用 LOAD 命令指定要装入的 XML 数据的位置、XML 数据的验证选项以及空格的处理方式。与 IMPORT 一样，可使用 xmlchar 和 xmlgraphic 文件修订符来对已装入 XML 数据指定编码特征。

导出 XML 数据

可以从包括一个或多个 XML 数据类型列的表中导出数据。导出的 XML 数据存储在与包含导出的关系数据的主数据文件不同的位置。导出的主数据文件中用 XML 数据说明符（XDS）表示关于每个导出的 XML 文档的信息。XDS 是一个字符串，它指定存储 XML 文档的系统文件的名称、此文件内 XML 文档的准确位置和长度以及用于验证 XML 文档的 XML 模式。

可以使用 EXPORT 命令的 XMLFILE、XML TO 和 XMLSAVESCHEMA 参数来指定关于如何存储导出的 XML 文档的详细信息。xmlinsepfiles、xmlnodeclaration、xmlchar 和 xmlgraphic 文件类型修饰符允许您指定关于导出的 XML 数据的存储位置和编码的更多详细信息。

有关移动 XML 数据的重要注意事项

以下是导入或导出 XML 数据时要谨记的一些注意事项:

- 导出的 XML 数据始终存储在与包含导出的关系数据的主数据文件不同的位置。
- 缺省情况下，EXPORT 实用程序采用 Unicode 写入 XML 数据。可以使用 xmlchar 文件类型修饰符来采用字符代码页写入 XML 数据。xmlgraphic 文件类型修饰符指定采用图形代码页（无论应用程序代码页是什么，图形代码页都是 UTF-16）来写入 XML 数据。

- 从版本 9.5 开始，XML 数据可存储在非 Unicode 数据库中，原因是插入前该数据将从数据库代码页转换为 UTF-8。为避免 XML 解析期间可能引入替换字符，要插入的字符数据应仅由数据库代码页中包含的代码点组成。将 enable_xmlchar 配置参数设置为 no 会阻止在 XML 解析期间插入字符数据类型，从而限制对未进行代码页转换的数据类型执行插入，如 BIT DATA、BLOB 或 pureXML™。
- 对于 IMPORT 和 LOAD 实用程序，除非要导入的 XML 文档包含的声明标记中包括编码属性，否则假定此文档采用 Unicode。可以使用 xmlchar 文件类型修饰符来指示要导入的 XML 文档采用字符代码页编码，而 xmlgraphic 文件类型修饰符指示要导入的 XML 文档采用 UTF-16编码。
- 对于 IMPORT 和 LOAD 实用程序，将拒绝包含结构不当的文档的行。
- 如果对 IMPORT 实用程序或 LOAD 实用程序指定了 XMLVALIDATE 选项，那么将针对其匹配模式验证成功的文档插入到表中时，会使用模式信息注释这些文档。如果行中包含的文档针对其匹配模式验证失败，那么将拒绝这些行。
- 可以使用具有 XQuery 规范的 EXPORT 实用程序来导出并非结构良好的 XML 文档的查询和 XPath 数据模型（XDM）实例。但是，不能直接将导出的结构不当的 XML 文档导入到 XML 列中，这是因为使用 XML 数据类型定义的列只能包含完整的 XML 文档。
- 如果要收集统计信息，那么 **CPU_PARALLELISM** 会在装入期间降至 1。
- XML 装入操作需要使用共享排序内存才能继续。因此，需要启用 **SHEAPTHRES_SHR** 或 **INTRA_PARALLEL** 或打开连接集中器。注意，缺省情况下 **SHEAPTHRES_SHR** 设置为某个值，所以会在缺省配置中提供共享排序内存。
- 装入表包含 XML 列的表中的操作不能指定 SOURCEUSEREXIT 选项、**SAVECOUNT** 参数或 anyorder 文件类型修饰符。
- 与 LOB 文件一样，XML 文件必须驻留在服务器端。

导入和导出时的 LOB 和 XML 文件行为

LOB 和 XML 文件共用导入和导出数据时可使用的一些行为和功能。

导出 导出数据时，如果使用 LOBS TO 选项指定了一个或多个 LOB 路径，那么 EXPORT 实用程序将循环使用这些 LOB 路径，以便将每个连续的 LOB 值写入相应的 LOB 文件。同样，如果使用 XML TO 选项指定了一个或多个 XML 路径，那么 EXPORT 实用程序将循环使用这些 XML 路径，以便将每个连续的 XQuery 和 XPath 数据模型（XDM）实例写入相应的 XML 文件。缺省情况下，LOB 值和 XDM 实例与导出的关系数据将写入同一路径。除非设置了 LOBSINSEPFILES 或 XMLINSEPFILES 文件类型修饰符，否则 LOB 文件和 XML 文件都可以有多个值并置至同一文件。

LOBFILE 选项提供了一种方法来指定 EXPORT 实用程序生成的 LOB 文件的基本名称。同样，XMLFILE 选项也提供了一种方法来指定 EXPORT 实用程序生成的 XML 文件的基本名称。缺省 LOB 文件基本名称是导出的数据文件的名称，其扩展名为 .lob。缺省 XML 文件基本名称是导出的数据文件的名称，其扩展名为 .xml。因此，导出的 LOB 文件或 XML 文件的全名由基本名称、接着是填满为三位数的编号扩展名以及 .lob 或 .xml 扩展名组成。

导入 导入数据时，LOB 位置说明符（LLS）与 XML 目标列兼容，而 XML 数据说明符（XDS）与 LOB 目标列兼容。如果未指定 LOBS FROM 选项，那么假定

要导入的 LOB 文件与输入关系数据文件位于同一路径中。同样，如果未指定 XML FROM 选项，那么假定要导入的 XML 文件与输入关系数据文件位于同一路径中。

导出示例

在以下示例中，所有 LOB 值将写入文件 /mypath/t1export.del.001.lob，而所有 XDM 实例将写入文件 /mypath/t1export.del.001.xml:

```
EXPORT TO /mypath/t1export.del OF DEL MODIFIED BY LOBSINFILE
SELECT * FROM USER.T1
```

在以下示例中，第一个 LOB 值将写入文件 /lob1/t1export.del.001.lob，第二个 LOB 值将写入文件 /lob2/t1export.del.002.lob，第三个 LOB 值将附加至 /lob1/t1export.del.001.lob，第四个 LOB 值将附加至 /lob2/t1export.del.002.lob，以此类推:

```
EXPORT TO /mypath/t1export.del OF DEL LOBS TO /lob1,/lob2
MODIFIED BY LOBSINFILE SELECT * FROM USER.T1
```

在以下示例中，第一个 XDM 实例将写入文件 /xml1/xmlbase.001.xml，第二个 XDM 实例将写入文件 /xml2/xmlbase.002.xml，第三个 XDM 实例将写入 /xml1/xmlbase.003.xml，第四个 XDM 实例将写入 /xml2/xmlbase.004.xml，以此类推:

```
EXPORT TO /mypath/t1export.del OF DEL XML TO /xml1,/xml2 XMLFILE xmlbase
  MODIFIED BY XMLINSEPFILES SELECT * FROM USER.T1
```

导入示例

对于包含单个 XML 列的“mytable”表和以下 IMPORT 命令:

```
IMPORT FROM myfile.del of del LOBS FROM /lobpath XML FROM /xmlpath
MODIFIED BY LOBSINFILE XMLCHAR replace into mytable
```

如果“myfile.del”包含以下数据:

```
mylobfile.001.lob.123.456/
```

IMPORT 实用程序将尝试从文件 /lobpath/mylobfile.001.lob 中文件偏移量为 123 处开始导入 XML 文档（其长度将为 456 字节）。

由于值由 LOB 位置说明符（LLS）而不是 XML 数据说明符（XDS）引用，因此假定“mylobfile.001.lob”文件位于 LOB 路径而不是 XML 路径中。

由于指定了 XMLCHAR 文件类型修饰符，因此假定文档采用字符代码页编码。

XML 数据说明符

使用 EXPORT、IMPORT 和 LOAD 实用程序移动的 XML 数据必须存储在与主数据文件分开的文件中。主数据文件中用 XML 数据说明符（XDS）表示 XML 数据。

XDS 是表示为 XML 标记（其名称是“XDS”）的字符串，它具有用于描述关于列中实际 XML 数据的信息的属性；这种信息涉及包含实际 XML 数据的文件名，以及该文件内 XML 数据的偏移量和长度。下面描述了 XDS 的属性。

FIL 包含 XML 数据的文件的名称。

OFF FIL 属性所指定的文件中 XML 数据的字节偏移量（其中偏移量从 0 开始）。

LEN FIL 属性所指定的文件中 XML 数据的长度（以字节计）。

SCH 用于验证此 XML 文档的 XML 模式的标准 SQL 标识。SQL 标识的模式和名称部分分别作为“OBJECTSCHEMA”和“OBJECTNAME”值存储在与此 XML 模式对应的 SYSCAT.XSROBJECTS 目录表的行中。

XDS 在数据文件中解释为字符字段，并且遵循文件格式的字符列解析行为。例如，对于定界 ASCII 文件格式（DEL），如果字符定界符出现在 XDS 中，那么该字符定界符必须加倍。属性值内的特殊字符（<、>、&、’ 和 ″）必须始终转义。区分大小写的对象名必须放在 " 字符实体之间。

示例

考虑值为 abc&"def".del 的 FIL 属性。要将此 XDS 包括在定界 ASCII 文件（其中字符定界符为 ″ 字符），必须使用两个 ″ 并且特殊字符必须转义。

```
<XDS FIL=""abc&amp;&quot;def&quot;.del"" />
```

以下示例显示 XDS 出现在定界 ASCII 数据文件中时的样式。XML 数据存储在 xmldocs.xml.001 文件中字节偏移量从 100 开始的位置，其长度为 300 字节。因为此 XDS 位于用双引号定界的 ASCII 文件中，所以 XDS 标记本身包含的双引号必须加倍。

```
"<XDS FIL = ""xmldocs.xml.001"" OFF=""100"" LEN=""300"" />"
```

以下示例显示标准 SQL 标识 ANTHONY.purchaseOrderTest。在 XDS 中，区分大小写的标识部分必须放在 " 字符实体间:

```
"<XDS FIL='/home/db2inst1/xmlload/a.xml' OFF='0' LEN='6758'
   SCH='ANTHONY.&quot;purchaseOrderTest&quot;' />"
```

查询和 XPath 数据模型

可通过使用以 SQL 提供的 XQuery 函数或者通过直接调用 XQuery 来访问数据库表中的 XML 数据。查询和 XPath 数据模型（XDM）的实例可能是格式良好的 XML 文档、节点序列、原子值序列或节点与原子值的任意组合。

可通过 EXPORT 命令将各个 XDM 实例写入一个或多个 XML 文件。

第 2 部分 附录

附录 A. import 和 load 实用程序之间的差别

下表对 DB2 的 load 实用程序与 import 实用程序之间的重要区别作了总结。

import 实用程序	load 实用程序
移动大量数据时速度较慢。	移动大量数据时，由于 load 实用程序将格式化的页直接写入数据库，所以速度比 import 实用程序快。
限制使用分区内并行性。只能通过在 ALLOW WRITE ACCESS 方式下并发调用 import 实用程序来实现分区内并行性。	可使用分区内并行性。通常，这需要对称多处理器（SMP）机器。
不支持 FASTPARSE。	支持 FASTPARSE，减少了对用户提供的数据进行的数据检查工作。
支持分层数据。	不支持分层数据。
能够创建 PC/IXF 格式的表、层次结构和索引。	表和索引必须存在。
不支持导入到具体化查询表中。	支持装入到具体化查询表中。
支持 WSF 格式。	不支持 WSF 格式。
不支持 BINARYNUMERICS。	支持 BINARYNUMERICS。
不支持 PACKEDDECIMAL。	支持 PACKEDDECIMAL。
不支持 ZONEDDECIMAL。	支持 ZONEDDECIMAL。
无法覆盖定义为 GENERATED ALWAYS 的列。	通过使用 `generatedoverride` 和 `identityoverride` 文件类型修饰符，可以覆盖 GENERATED ALWAYS 列。
支持导入到表、视图和昵称中。	仅支持装入到表中。
记录所有行。	执行最少的记录。
支持触发器。	不支持触发器。
如果导入操作被中断，并且指定了 *commitcount*，那么该表可供使用，并且将包含最后一次落实（COMMIT）之前已装入的行。用户可以重新启动导入操作，也可以按原样接受该表。	如果装入操作被中断，并且指定了 *savecount*，那么在重新启动装入操作、调用装入终止操作或者根据尝试执行装入操作前创建的备份映像复原表空间之前，该表将保持装入暂挂状态并且不可用。
所需空间量大概等于最大索引大小加上 10%。此空间是从数据库中的临时表空间中获取的。	所需空间量大概等于对该表定义的所有索引的大小之和，并且可能是此大小的两倍。此空间是从数据库中的临时空间中获取的。
在导入操作期间将验证所有约束。	load 实用程序将检查唯一性并计算生成列值，但必须使用 SET INTEGRITY 来检查所有其他约束。
在导入操作期间，将逐个地把键值插入到索引中。	对键值进行排序，装入数据后构建索引。
如果需要更新的统计信息，导入操作完成后必须运行 Runstats 实用程序。	如果正在替换表中的所有数据，那么可以在装入操作执行期间收集统计信息。
可以通过 DB2 Connect 导入到主机数据库中。	不能装入到主机数据库中。

import 实用程序	load 实用程序
导入文件必须在调用 import 实用程序的客户机上。	根据指定的选项，装入文件或管道可以在包含数据库的数据库分区上，也可以在所连接的调用 load 实用程序的远程客户机上。 **注**: 只能从服务器端读取 LOB 和 XML 数据。
不需要备份映像。由于 import 实用程序使用 SQL 插入，所以将记录活动，因此发生故障时不需要备份就可以恢复这些操作。	在装入操作执行期间，可以创建备份映像。

附录 B. export、import 和 load 实用程序使用的绑定文件

下表列示了绑定文件、其缺省隔离级别、使用它们的实用程序以及使用目的。

绑定文件（缺省隔离级别）	实用程序/目的
db2ueiwi.bnd（CS）	IMPORT/EXPORT。用来查询关于表列和索引的信息。
db2uexpm.bnd（CS）	EXPORT。用来从对导出操作指定的查询中进行访存。
db2uimpm.bnd（RS）	IMPORT。当使用了 INSERT、REPLACE 或 REPLACE_CREATE 选项时，用来将源数据文件中的数据插入到目标表中。 **注**：不推荐使用 IMPORT 命令的 CREATE 和 REPLACE_CREATE 选项，将来的发行版中可能会除去这两个选项。
db2uipkg.bnd（CS）	IMPORT。用来检查绑定选项。
db2uiici.bnd（RR）	IMPORT。当指定了 IXF CREATE 选项时，用来创建索引。
db2ucktb.bnd（CS）	装入。用来执行装入操作的一般初始化过程。
db2ulxld.bnd（CS）	装入。用来处理“从游标装入”操作期间提供的查询。
db2uigsi.bnd（在基于 UNIX 的系统上是 RS，在所有其他平台上是 RR）	IMPORT/EXPORT。用来删除索引和检查导入替换操作的引用约束，并且用来检索用于导出 IXF 文件的标识列信息。
db2uiict.bnd（RR）	IMPORT。当指定了 IXF CREATE 选项时，用来创建表。
db2uqtpd.bnd（RR）	IMPORT/EXPORT。用来执行分层表处理。
db2uqtnm.bnd（RR）	IMPORT。当指定了 IXF CREATE 选项时，用来执行分层表处理。
db2uimtb.bnd（RS）	IMPORT。用来执行导入操作的一般初始化过程。
db2uImpInsUpdate.bnd（RS）	IMPORT。当使用了 INSERT_UPDATE 选项时，用来将源数据文件中的数据插入到目标表中。无法与 INSERT BUF 选项绑定。

附录 C. 如何阅读语法图

在本书中，是按照如下所示定义的结构来描述语法的:

沿着主干的方向，按照从左到右、从上到下的顺序阅读语法图。

►►— 符号表示语法图的开头。

—► 符号表示语法紧接着下一行。

►— 符号表示语法紧接着上一行。

—►◄ 符号表示语法图的结尾。

语法段以 ├— 符号开头，以 —┤ 符号结尾。

必需项都显示在水平线（主路径）上。

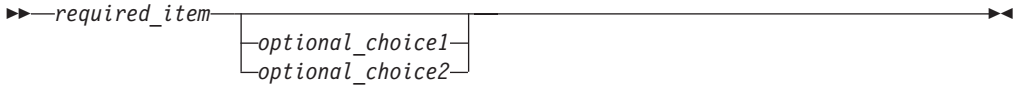
可选项显示在主路径下方。

如果某个可选项显示在主路径上方，那么在执行时该项将不起作用，仅用于提高可读性而已。

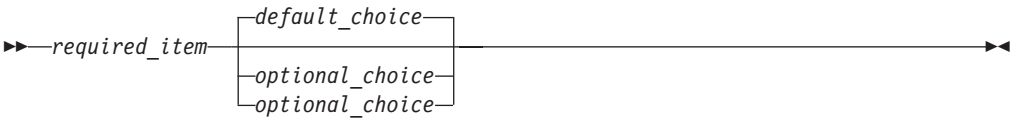
如果您可以从两项或多项中进行选择，那么它们会以堆叠的形式出现。

如果您*必须*选择其中一项，那么堆叠项中的一项应显示在主路径上。

如果可根据情况选择其中一项，那么所有堆叠项将显示在主路径下方。



如果其中一项是缺省值，那么该项将显示在主路径上方，而其余项将显示在主路径下方。



如果一个箭头从主干上方返回到左边，那么表示可以重复使用该项。在这种情况下，各个重复项之间必须用一个或多个空格隔开。

如果重复箭头中包含一个逗号，那么必须用逗号将各个重复项分隔开。

如果重复箭头位于堆叠项上方，那么表示您可以选择多个堆叠项，或者重复选择单个项。

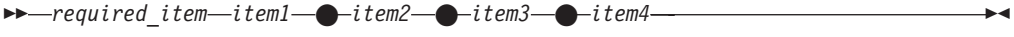
关键字都采用大写字母（例如，FROM）。它们必须按照所显示那样拼写。变量都采用小写字母（例如，column-name）。它们表示语法中由用户提供的名称或值。

如果显示了标点符号、圆括号、算术运算符或者其他这样的符号，那么必须将它们作为语法的一部分来输入。

有时，单个变量表示一个很大的语法段。例如，在下图中，parameter-block 变量表示标有 **parameter-block** 的整个语法段:

parameter-block:

“大项目符号”（●）之间的相邻段可以按任意顺序指定。



上图表示可以按任意顺序指定 item2 和 item3。下面两种情况都有效:

```
required_item item1 item2 item3 item4
required_item item1 item3 item2 item4
```

附录 D. 收集关于数据移动问题的数据

如果您在执行数据移动命令时遇到问题并且不能确定问题的原因，请收集您或 IBM 软件支持机构可用来诊断和解决问题的诊断数据。

- 要收集与 db2move 命令相关的问题的数据，请转至发出了此命令的目录。根据在此命令中指定的操作找到下列文件:
  - 对于 COPY 操作，查找名为 COPY.*timestamp*.ERR 和 COPYSCHEMA.*timestamp*.MSG 的文件。如果还指定了 LOAD_ONLY 或 DDL_AND_LOAD 方式，还需查找名为 LOADTABLE.*timestamp*.MSG 的文件。
  - 对于 EXPORT 操作，查找名为 EXPORT.out 的文件。
  - 对于 IMPORT 操作，查找名为 IMPORT.out 的文件。
  - 对于 LOAD 操作，查找名为 LOAD.out 的文件。
- 要收集与 EXPORT、IMPORT 或 LOAD 命令相关的问题的数据，确定命令是否包括 MESSAGES 参数。如果包括此参数，那么收集输出文件。如果您不指定其他目录和驱动器，那么这些实用程序使用当前目录和缺省驱动器作为目标。
- 要收集与 REDISTRIBUTE 命令相关的问题的数据，在 Linux 和 UNIX 上，查找名为“*databasename.database_partition_groupname.timestamp*”的文件；在 Windows 上，查找名为“*databasename.database_partition_groupname.date.time*”的文件。文件分别位于 *$HOME*/sqllib/db2dump 目录或 $DB2PATH\sqllib\redist 目录中，其中 *$HOME* 是实例所有者的主目录。

附录 E. DB2 技术信息概述

可以通过下列工具和方法获取 DB2 技术信息:

- DB2 信息中心
  - 主题（任务、概念和参考主题）
  - DB2 工具的帮助
  - 样本程序
  - 教程
- DB2 书籍
  - PDF 文件（可下载）
  - PDF 文件（在 DB2 PDF DVD 中）
  - 印刷版书籍
- 命令行帮助
  - 命令帮助
  - 消息帮助

注: DB2 信息中心主题的更新频率比 PDF 书籍或硬拷贝书籍的更新频率高。要获取最新信息，请安装可用的文档更新，或者参阅 ibm.com® 上的 DB2 信息中心。

可以在线访问 ibm.com 上的其他 DB2 技术信息，如技术说明、白皮书和 IBM Redbooks 出版物。访问位于以下网址的 DB2 信息管理软件库站点: http://www.ibm.com/software/data/sw-library/。

文档反馈

我们非常重视您对 DB2 文档的反馈。如果您想就如何改善 DB2 文档提出建议，请将电子邮件发送至 db2docs@ca.ibm.com。 DB2 文档小组会阅读您的所有反馈，但不能直接答复您。请尽可能提供具体的示例，这样我们才能更好地了解您所关心的问题。如果您要提供有关具体主题或帮助文件的反馈，请加上标题和 URL。

请不要用以上电子邮件地址与 DB2 客户支持机构联系。 如果您遇到文档不能解决的 DB2 技术问题，请与您当地的 IBM 服务中心联系以获得帮助。

硬拷贝或 PDF 格式的 DB2 技术库

下列各表描述 IBM 出版物中心（网址为 www.ibm.com/shop/publications/order）提供的 DB2 资料库。可以从 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 下载 PDF 格式的英文 DB2 版本 9.5 手册和已翻译的版本。

尽管这些表标识书籍有印刷版，但可能未在您所在国家或地区提供。

每次更新手册时，表单号都会递增。确保您正在阅读下面列示的手册的最新版本。

注: DB2 信息中心比 PDF 或硬拷贝书籍的更新频率高。

表 62. DB2 技术信息

书名	书号	是否提供印刷版
Administrative API Reference	SC23-5842-01	是
Administrative Routines and Views	SC23-5843-01	否
Call Level Interface Guide and Reference, Volume 1	SC23-5844-01	是
Call Level Interface Guide and Reference, Volume 2	SC23-5845-01	是
Command Reference	SC23-5846-01	是
《数据移动指南和参考》	S151-0617-01	是
《数据恢复及高可用性指南与参考》	S151-0619-01	是
《数据服务器、数据库和数据库对象指南》	S151-0612-01	是
《数据库安全性指南》	S151-0614-01	是
Developing ADO.NET and OLE DB Applications	SC23-5851-01	是
Developing Embedded SQL Applications	SC23-5852-01	是
Developing Java Applications	SC23-5853-01	是
Developing Perl and PHP Applications	SC23-5854-01	否
Developing User-defined Routines (SQL and External)	SC23-5855-01	是
Getting Started with Database Application Development	GC23-5856-01	是
《Linux 和 Windows 上的 DB2 安装和管理入门》	G151-0623-01	是
《国际化指南》	S151-0616-01	是
《消息参考，第 1 卷》	G151-0632-00	否
《消息参考，第 2 卷》	G151-0633-00	否
《迁移指南》	G151-0622-01	是
《Net Search Extender 管理和用户指南》	S151-0760-01	是
《分区和集群指南》	S151-0615-01	是
Query Patroller Administration and User's Guide	SC23-8507-00	是
《IBM 数据服务器客户机快速入门》	G151-0625-01	否
《DB2 服务器快速入门》	G151-0624-01	是
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC23-8508-01	是

表 62. DB2 技术信息 (续)

书名	书号	是否提供印刷版
SQL Reference, Volume 1	SC23-5861-01	是
SQL Reference, Volume 2	SC23-5862-01	是
《系统监视器指南和参考》	S151-0618-01	是
《故障诊断指南》	G151-0621-01	否
《调整数据库性能》	S151-0613-01	是
《Visual Explain 教程》	S151-0634-00	否
《新增内容》	S151-0629-01	是
Workload Manager Guide and Reference	SC23-5870-01	是
《pureXML 指南》	S151-0630-01	是
《XQuery 参考》	S151-0631-01	否

表 63. 特定于 DB2 Connect 的技术信息

书名	书号	是否提供印刷版
《DB2 Connect 个人版快速入门》	G151-0627-01	是
《DB2 Connect 服务器快速入门》	G151-0628-01	是
《DB2 Connect 用户指南》	S151-0626-01	是

表 64. Information Integration 技术信息

书名	书号	是否提供印刷版
Information Integration: Administration Guide for Federated Systems	SC19-1020-01	是
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	是
《Information Integration: 联合数据源配置指南》	S151-0468-00	否
《Information Integration: SQL 复制指南和参考》	S151-0475-00	是
Information Integration: Introduction to Replication and Event Publishing	SC19-1028-01	是

订购印刷版的 DB2 书籍

如果您需要印刷版的 DB2 书籍，可以在许多（但不是所有）国家或地区在线购买。无论何时都可以从当地的 IBM 代表处订购印刷版的 DB2 书籍。请注意，*DB2 PDF 文档* DVD 上的某些软拷贝书籍没有印刷版。例如，*DB2 消息参考*的任何一卷都没有提供印刷版书籍。

只要支付一定费用，就可以从 IBM 获取 DB2 PDF 文档 DVD，该 DVD 包含许多 DB2 书籍的印刷版。 根据您下订单的位置，您可能能够从 IBM 出版物中心在线订购书籍。如果在线订购在您所在国家或地区不可用，您总是可以从当地的 IBM 代表处订购印刷版 DB2 书籍。注意，并非 DB2 PDF 文档 DVD 上的所有书籍都有印刷版。

注：最新最完整的 DB2 文档保留在网址如下的 DB2 信息中心中：http://publib.boulder.ibm.com/infocenter/db2luw/v9r5。

要订购印刷版的 DB2 书籍：

- 要了解您是否可从所在国家或地区在线订购印刷版的 DB2 书籍，可查看 IBM 出版物中心站点，网址为：http://www.ibm.com/shop/publications/order。 必须先选择国家、地区或语言才能访问出版物订购信息，然后再按照针对您所在位置的订购指示信息进行订购。
- 要从当地的 IBM 代表处订购印刷版的 DB2 书籍：
  1. 从下列其中一个 Web 站点找到当地代表处的联系信息：
     - IBM 全球联系人目录，网址为 www.ibm.com/planetwide
     - IBM 出版物 Web 站点，网址为 http://www.ibm.com/shop/publications/order。必须先选择国家、地区或语言才能访问对应您的所在地的出版物主页。在此页面中访问“关于此站点”链接。
  2. 请在致电时说明您想订购 DB2 出版物。
  3. 请向您当地的代表提供想要订购的书籍的书名和书号。有关书名和书号的信息，请参阅第 405 页的『硬拷贝或 PDF 格式的 DB2 技术库』。

从命令行处理器显示 SQL 状态帮助

DB2 返回描述 SQL 语句执行结果的 SQLSTATE。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

要调用 SQL 状态帮助，打开命令行处理器并输入：

```
? sqlstate or ? class code
```

其中，*sqlstate* 表示有效的 5 位 SQL 状态，*class code* 表示该 SQL 状态的前 2 位。

例如，`? 08003` 显示 08003 SQL 状态的帮助，而 `? 08` 显示 08 类代码的帮助。

访问不同版本的 DB2 信息中心

对于 DB2 版本 9.5 主题，DB2 信息中心 URL 为http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/

对于 DB2 版本 9 主题，DB2 信息中心 URL 为 http://publib.boulder.ibm.com/infocenter/db2luw/v9/

对于 DB2 版本 8 主题，请访问以下版本 8 信息中心 URL：http://publib.boulder.ibm.com/infocenter/db2luw/v8/

在 DB2 信息中心中以您的首选语言显示主题:

DB2 信息中心尝试以您在浏览器首选项中指定的语言显示主题。如果未提供主题的首选语言翻译版本，那么 DB2 信息中心将显示该主题的英文版。

- 要在 Internet Explorer 浏览器中以您的首选语言显示主题:
  1. 在 Internet Explorer 中，单击**工具 —> Internet 选项 —> 语言...**按钮。“语言首选项”窗口打开。
  2. 确保您的首选语言被指定为语言列表中的第一个条目。
     - 要将新语言添加至列表，单击**添加...** 按钮。

       注: 添加语言并不能保证计算机具有以首选语言显示主题所需的字体。
     - 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。
  3. 清除浏览器高速缓存然后刷新页面以便以首选语言显示 DB2 信息中心。
- 要在 Firefox 或 Mozilla 浏览器中以首选语言显示主题:
  1. 在**工具 —> 选项 —> 高级**对话框中的**语言**部分中选择按钮。“语言”面板将显示在“首选项”窗口中。
  2. 确保您的首选语言被指定为语言列表中的第一个条目。
     - 要将新语言添加至列表，单击**添加...** 按钮以从“添加语言”窗口中选择一种语言。
     - 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。
  3. 清除浏览器高速缓存然后刷新页面以便以首选语言显示 DB2 信息中心。

在某些浏览器和操作系统组合上，可能还必须将操作系统的区域设置更改为您选择的语言环境和语言。

更新安装在您的计算机或内部网服务器上的 DB2 信息中心

如果已经在本地安装了 DB2 信息中心，那么您可以从 IBM 获取文档更新并安装。

更新在本地安装的 DB2 信息中心要求您:

1. 停止计算机上的 DB2 信息中心，然后以独立方式重新启动信息中心。如果以独立方式运行信息中心，那么网络上的其他用户将无法访问信息中心，因而您可以应用更新。非管理和非根 DB2 信息中心始终以独立方式运行。。
2. 使用“更新”功能部件来查看可用的更新。如果有您希望安装的更新，那么请使用“更新”功能部件来获取并安装这些更新。

   注: 如果您的环境要求在一台未连接至因特网的机器上安装 DB2 信息中心更新，那么必须使用一台已连接至因特网的机器将更新站点镜像至本地文件系统并安装 DB2 信息中心。如果网络中有许多用户将安装文档更新，那么可以通过在本地也为更新站点建立镜像并为更新站点创建代理来缩短每个人执行更新所需要的时间。
   如果提供了更新包，请使用“更新”功能部件来获取这些更新包。但是，只有在独立方式下才能使用“更新”功能部件。
3. 停止独立信息中心，然后在计算机上重新启动 DB2 信息中心。

注: 在 Windows Vista 上，必须以管理员身份才能运行下面所列示的命令。要启动具有所有管理员特权的命令提示符或图形工具，右键单击快捷方式，然后选择**以管理员身份运行**。

要更新安装在您的计算机或内部网服务器上的 DB2 信息中心:

1. 停止 DB2 信息中心。
   - 在 Windows 上，单击**开始 → 控制面板 → 管理工具 → 服务**。右键单击**DB2 信息中心**服务，并选择**停止**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv95 stop
     ```
2. 以独立方式启动信息中心。
   - 在 Windows 上:
     a. 打开命令窗口。
     b. 浏览至信息中心的安装位置。缺省情况下，DB2 信息中心安装在 <Program Files>\IBM\DB2 Information Center\Version 9.5 目录中，其中 <Program Files> 表示 Program Files 目录的位置。
     c. 从安装目录浏览至 doc\bin 目录。
     d. 运行 help_start.bat 文件:

        ```
        help_start.bat
        ```
   - 在 Linux 上:
     a. 浏览至信息中心的安装位置。缺省情况下，DB2 信息中心安装在 /opt/ibm/db2ic/V9.5 目录中。
     b. 从安装目录浏览至 doc/bin 目录。
     c. 运行 help_start 脚本:

        ```
        help_start
        ```

   系统缺省 Web 浏览器将启动以显示独立信息中心。
3. 单击**更新**按钮（ ）。在信息中心的右边面板上，单击**查找更新**。 将显示现有文档的更新列表。
4. 要启动安装进程，请检查您要安装的选项，然后单击**安装更新**。
5. 在安装进程完成后，请单击**完成**。
6. 要停止独立信息中心，请执行下列操作:
   - 在 Windows 上，浏览至安装目录的 doc\bin 目录并运行 help_end.bat 文件:

     ```
     help_end.bat
     ```

     注: help_end 批处理文件包含安全地终止使用 help_start 批处理文件启动的进程所需的命令。不要使用 Ctrl-C 或任何其他方法来终止 help_start.bat。
   - 在 Linux 上，浏览至安装目录的 doc/bin 目录并运行 help_end 脚本:

     ```
     help_end
     ```

     注: help_end 脚本包含安全地终止使用 help_start 脚本启动的进程所需的命令。不要使用任何其他方法来终止 help_start 脚本。
7. 重新启动 DB2 信息中心。

- 在 Windows 上，单击**开始 → 控制面板 → 管理工具 → 服务**。右键单击 **DB2 信息中心**服务，并选择**启动**。
- 在 Linux 上，输入以下命令:

  ```
  /etc/init.d/db2icdv95 start
  ```

更新后的 DB2 信息中心将显示新的主题和更新后的主题。

DB2 教程

DB2 教程帮助您了解 DB2 产品的各个方面。这些课程提供了逐步指示信息。

开始之前

可从信息中心查看 XHTML 版的教程: http://publib.boulder.ibm.com/infocenter/db2help/ 。

某些课程使用了样本数据或代码。有关其特定任务的任何先决条件的描述，请参阅教程。

DB2 教程

要查看教程，请单击标题。

*《pureXML 指南》*中的**『pureXML』**
设置 DB2 数据库以存储 XML 数据以及如何对本机 XML 数据存储执行基本操作。

*《Visual Explain 教程》*中的**『Visual Explain』**
使用 Visual Explain 来分析、优化和调整 SQL 语句以获取更好的性能。

DB2 故障诊断信息

很多故障诊断和问题确定信息可帮助您使用 DB2 产品。

DB2 文档
故障诊断信息可在 DB2 信息中心的“DB2 故障诊断指南”或“支持和故障诊断”部分找到。可在该处找到有关如何使用 DB2 诊断工具和实用程序隔离和找出问题的信息、某些最常见问题的解决方案以及有关如何解决使用 DB2 产品时可能遇到的问题的建议。

DB2 技术支持 Web 站点
如果您遇到了问题并且想要获取查找可能的原因和解决方案的帮助，请参阅 DB2 技术支持 Web 站点。该“技术支持”站点具有指向最新 DB2 出版物、技术说明、授权程序分析报告（APAR 或错误修订）、修订包和其他资源的链接。可搜索此知识库并查找问题的可能解决方案。

访问位于以下网址的 DB2 技术支持 Web 站点: http://www.ibm.com/software/data/db2/udb/support.html

条款和条件

如果符合以下条款和条件，那么授予您使用这些出版物的准用权。

个人使用: 只要保留所有的专有权声明，您就可以为个人、非商业使用复制这些出版物。未经 IBM 明确同意，您不可以分发、展示或制作这些出版物或其中任何部分的演绎作品。

商业使用: 只要保留所有的专有权声明，您就可以仅在企业内复制、分发和展示这些出版物。未经 IBM 明确同意，您不可以制作这些出版物的演绎作品，或者在您的企业外部复制、分发或展示这些出版物或其中的任何部分。

除非本准用权中有明确授权，不得把其他准用权、许可或权利（无论是明示的还是暗含的）授予其中包含的出版物或任何信息、数据、软件或其他知识产权。

当使用这些出版物损害了 IBM 的利益，或者根据 IBM 的规定，未正确遵守上述指导说明时，那么 IBM 保留自主决定撤销本文授予的准用权的权利。

您不可以下载、出口或再出口本信息，除非完全遵守所有适用的法律和法规，包括所有美国出口法律和法规。

IBM 对这些出版物的内容不作任何保证。这些出版物“按现状”提供，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的关于适销和适用于某种特定用途的保证。

附录 F. 声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，那么由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本文档可能提供非 IBM Web 站点和资源的链接或引用。IBM 对于任何非 IBM Web 站点或第三方资源不作任何声明、保证或其他承诺，即使本文档可能引用了这些 Web 站点或第三方资源，或者可从本文档访问或链接到这些 Web 站点或第三方资源。到某个非 IBM Web 站点的链接并不意味着 IBM 认可此类 Web 站点的内容或使用或其所有者。此外，IBM 不是您与任何第三方签署协议的任何交易的一方，也不对任何交易负责，即使您从某个 IBM 站点了解到此类第三方或使用到此类第三方的链接时亦如此。因此，您需要承认并同意，IBM 不对此类外部站点或资源的可用性负责，也不对可从那些站点或资源上获得的任何内容、服务、产品或其他资料承担任何责任或义务。第三方提供的任何软件须遵守该软件随附的许可证的条款和条件。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息可能包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

版权许可:

本信息可能包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（*贵公司的名称*）（*年份*）. 此部分代码是根据 IBM 公司的样本程序衍生出来的。© Copyright IBM Corp. *（输入年份）*. All rights reserved.

商标

下列术语是 International Business Machines Corporation 在美国和/或其他国家或地区的商标或注册商标。

pureXML
Informix
DB2
AT
WebSphere
DB2 Connect
z/OS
System i
Lotus
Tivoli
MVS
OS/400
ibm.com
DS8000
1-2-3
TotalStorage
REXX
AIX
OS/390
DB2 Universal Database
Redbooks
IBM
System/370
DRDA
System Storage
DS6000
Enterprise Storage Server

下列术语是其他公司的商标或注册商标。

- Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。
- Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和/或其他国家或地区的商标。
- UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。
- Intel 是 Intel 公司或其子公司在美国和其他国家或地区的注册商标。
- Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

[B]

[C]

[D]

[E]

[F]

[G]

[H]

[J]

[K]

[L]

[M]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

A

C

D

E

F

G

I

K

L

M

N

P

R

S

T

U

V

W

X

[特别字符]

中国印刷

S151-0617-01

Spine information:

DB2 版本 **9.5 Linux** 版、**UNIX** 版和 **Windows** 版

数据移动指南和参考